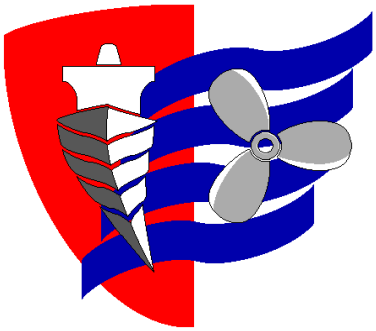
ESCUELA TÉCNICA SUPERIOR DE NÁUTICA

UNIVERSIDAD DE CANTABRIA



Trabajo Fin de Master.

Planeador submarino (AUV GLIDER), electrónica, control y navegacion.

Underwater GLIDER (AUV), electronics, control and navigation.

Para acceder al Título de Master en

INGENIERÍA MARINA

Autor: David Espinosa García

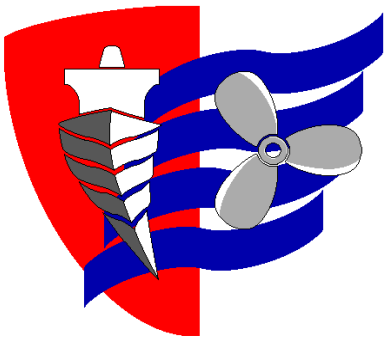
Director: Alfredo Girón Portilla

Codirector: Jesús María Villar González

Septiembre – 2020

ESCUELA TÉCNICA SUPERIOR DE NÁUTICA

UNIVERSIDAD DE CANTABRIA



Trabajo Fin de Master

Planeador submarino (AUV GLIDER), electrónica, control y navegación.

Underwater GLIDER (AUV), electronics, control and navigation.

Para acceder al Título de Master en

INGENIERÍA MARINA

INDICE

Índice de Ilustraciones

Resumen.

Este trabajo es el estudio y puesta en operación de un sistema de navegación para un AUV GLIDER o planeador submarino.

Es el desarrollo de una parte de un producto ya diseñado o existente, al que hay que dotarle de la operatividad necesaria para navegar.

El AUV GLIDER no tiene partes móviles externas y su maniobrabilidad y propulsión se basan en la flotabilidad. Esta fuerza puede ser variada a voluntad tanto en magnitud como su línea de acción. Eso crea una aerodinámica similar a la de los aero-planeadores de ala fija, en el ámbito submarino.

En el trabajo se trata la descripción de los motores, sensores, estrategia y lógica de control, así como los diversos elementos de electrónica y electricidad necesario para implementar las funciones.

Se contempla la navegación autónoma, y la capacidad de supervivencia ante incidentes, estableciendo estrategia orientadas a la seguridad de la nave.

El estudio es el desarrolla de la funcionalidad de navegación descrito del TFG “Planeador submarino (AUV GLIDER) diseño y construcción” de este mismo autor, donde ya quedo resuelto la definición de la estructura y la estanqueidad.

1. Gestión integral de la navegación y de la seguridad.

Dado el medio en el que ha de mantenerse el AUV, hemos considerado que la seguridad debe orientarse a navegar según el modo previsto, regresar a 'casa' si las cosas se ponen mal, y 'sobrevivir' si todo lo previsto falla y debe ser rescatado.

Vamos a hacer un resumen de los objetivos, desde la situación más sencilla a la más grave:

- El primer objetivo en nivel simple de seguridad sería la gestión cotidiana de la navegación, corregir desviación de rumbo, control de trimado, de velocidad, mantenerse dentro de las limitaciones de profundidad, control de consumo y energía disponible, etc.

- El siguiente nivel de seguridad, sería que el AUV tuviese capacidad de gestionar de modo autónomo diversas vicisitudes indeseables durante la navegación: alcanzar profundidades no deseables, alcanzar temperaturas excesivas o parámetros no deseables en los sistemas de control, etc. Esas son circunstancias que admiten un control ordenado de la emergencia, y mantienen al AUV activo. El sistema de gestión debe pasar automáticamente al modo 'vuelta a casa' (con la energía disponible),

- El siguiente nivel de seguridad sería posibilitar ser rescatado en una situación de pérdida total de control. Básicamente pérdida de energía, o fallo del sistema de navegación. Este sería para nosotros la peor situación de las previstas, y la respuesta es intentar volver a la superficie: que el sistema pase a flotabilidad positiva, en modo rescate.

- El último nivel de seguridad sería no inundarse, ni sucumbir ante la presión hidrostática. Un colapso estructural o una inundación masiva

solo podrían ser contrarrestados con un diseño del AUV compartimentado en zonas independientes, de modo que el fallo en alguna de ellas no suponga el fallo de las demás. Así ocurre en los sumergibles ‘reales’ con responsabilidad. No obstante, la complejidad estructural que eso supondría para nuestro caso nos lleva a renunciar directamente a esa seguridad. Nos queda centrarnos en los otros aspectos de la seguridad. Esta filosofía es realista, y también se aplica a sumergibles turísticos de pasajeros, donde la disposición estructural no contempla un compartimentado seguro, y en su lugar la seguridad se enfoca hacia otros aspectos ‘equivalentes’: operar en zonas con fondo a profundidad limitada, inspección periódica de la estructura, etc.

2. El control de la nave a manos de Arduino.

Todo el control del AUV está basado en una pequeña placa electrónica llamada **Arduino**, que contiene un microprocesador, con posibilidad de ser programado en un lenguaje semejante al C, dentro de un IDE (entorno integrado de desarrollo de programación) propio que facilita la edición de código, la compilación y la carga de programas en el procesador, así como monitorizar las comunicaciones con el Arduino vía puerto serie.

Su diseño es conocido, dentro de la modalidad Open Hardware - open Source, y existe mucha información: bibliografía, cursos, tutoriales, videos, etc., para hacer todo tipo de proyectos de base electrónica sobre esa plataforma.

https://es.wikipedia.org/wiki/Arduino
https://en.wikipedia.org/wiki/Arduino

La placa posee una notable cantidad de pines a través de los cuales se puede comunicar con el entorno exterior. Los pines se clasifican en varios tipos: analógicos o digitales, de entrada, o de salida, de comunicaciones.

https://aprendiendoarduino.wordpress.com/2016/06/27/arduino-uno-a-fondo-mapa-de-pines-2/

Cada pin digital de entrada o de salida, permite comunicarse con el exterior según la lógica TTL, con nivel de tensión 0-5V.

Cada pin analógico de entrada o salida, puede recibir o mandar una señal de tensión entre 0 y 5V, que se mapea internamente en un valor entero en los rangos 0-255, o 0-1023, según sea el conversor analógico-digital usado de 8 o de 10 bits.

A través de los pines de entrada, la unidad de control puede recibir datos de sensores de tipos diversos: presión, temperatura, aceleración, actitud 3D,

tensión, corriente, velocidad de giro de motores, etc.

Igualmente, a través de los pines de salida, la unidad de control puede manejar actuadores de diversos tipos: relés, reguladores de velocidad de motores, servos, etc.

Con la programación adecuada, se puede disponer una colección de rutinas que haga posible la navegación autónoma del AUV, y la ejecución de misiones según un guion establecido.

Ese conjunto de rutinas constituye la 'forma de pensar' y de 'mandar' que tiene el Arduino para gobernar la nave.

Pretendemos conseguir formar un bloque, que pueda ser diferenciado del resto de la electromecánica del AUV. Que el cambio hipotético a otra plataforma CPU no sea una dificultad. En ese bloque hay que tratar

- Tipos de comunicaciones entre los dispositivos: I2C, UART, ISP…

- Necesidad de adaptadores de niveles lógicos de tensión,

- Gestión de interrupciones. Es la base para tratar las alarmas y emergencias

- Recursos hardware vs software para tareas como comunicaciones serie, interrupciones, etc.

- Optimización de la velocidad de transmisión y recursos de CPU.

3. La tripulación del AUV

Si el “mando” de la nave es Arduino, la “tripulación” son unos Shields, actuadores, y sensores.

Un shield es una placa electrónica dotada de unos componentes adecuados para hacer algún trabajo concreto, en coordinación con la placa Arduino, y ampliando las posibilidades de esta. Los shields se ‘enchufan’ a algunos de los pines del Arduino. Los pines no usados por el Shield siguen estando disponibles para que el Arduino los pueda usar en otras tareas ajenas a ese shield, según su programación. Nosotros vamos a usar 2 shields:

- MotorShield, capaz de controlar motoresDC, servos y motores stepper,

- Data Logger Shield, capaz de grabar información en una tarjeta SD, y dispone de un reloj de tiempo real (RTC), que permite asociar a cada dato grabado la hora a la que se registra.

Se podrían disponer otros Shields adicionales, pero hay que asegurarse de la compatibilidad de los pines usados por todos ellos. Existe información específica sobre ese tema:

- http://shieldlist.org/
- https://learn.adafruit.com/adafruit-shield-compatibility/overview
- http://shieldlist.org/adafruit/motor
- http://shieldlist.org/adafruit/logger

4. Las comunicaciones entre Capitán y tripulación (Arduino y los dispositivos conectados)

Inevitablemente debemos descender un poco más al detalle de la placa Arduino y sus propiedades, aunque intentaremos ser breves.

Las comunicaciones entre la placa y los diversos elementos pueden hacerse de varios modos. Nosotros usaremos básicamente 3 variantes:

- I2C, (también denominado IIC, o TWI) es un protocolo de comunicación síncrona, que usa 2 cables, en una arquitectura maestro-esclavos. Cada esclavo tiene una dirección identificativa.
    - La arquitectura I2C es una alternativa a otro protocolo (el ISP, que tiene algunas desventajas para nosotros, más cables, etc.). No obstante, pueden usarse ambos protocolos sobre la misma placa en la misma aplicación.
    - Existen 2 modos dentro de Arduino para manejar I2C:
        - Por hardware, admite más velocidad y consume menos recursos de máquina, pero en cada placa hay solo unos pocos pines para ese canal. Requiere usar la librería Wire.h
        - Por software, pueden usarse todos lo pines de la placa, pero consume más recursos de procesador y es más lento.
- Uso de los pines individuales del Arduino.
- UART. protocolo serie asíncrono, también se puede usar complementando lo anterior.

Más detalles sobre las comunicaciones del Arduino en:

- https://www.drouiz.com/blog/2018/06/25/uart-vs-spi-vs-i2c-diferencias-entre-protocolos/

- https://aprendiendoarduino.wordpress.com/2014/11/18/tema-6-comunicaciones-con-arduino-4/ Muy completa descripción, en español.

- https://www.adafruit.com/product/757, convertidor bidireccional de niveles lógicos de tensión (3.3v – 5V, etc) entre dispositivos que lo requieran. Todas las comunicaciones con varios esclavos pueden pasar sus 2 líneas DTA, CLK por dos únicos canales bidireccionales del convertidor. Basta un convertidos por cada bus, y en un mismo bus caben hasta 112 esclavos.

  Más adelante veremos que algunos de nuestros esclavos requerirán ese convertidor de niveles:

  - Módulo BT, su lógica trabaja en 3.3v, aunque la placa admite hasta 5V. Cuando se comunica vía UART con el Arduino, hay que reducir la tensión de las señales entrante en el BT, si son a más de 3.3V.

  - Unidad IMU-MPU inercial, su lógica trabaja en 3.3v

  - También se puede encontrar otros tipos.

5. Funciones motorizadas del AUV.

Nuestro AUV no tiene partes externas móviles (salvo la parte expuesta del émbolo de flotabilidad). La propulsión se logra por la flotabilidad de la cámara de proa. Y el cambio de rumbo se consigue moviendo transversalmente un peso interno.

Para cada una de esas funciones, propulsión y gobierno, se empleará un motor eléctrico.

- En el caso de la propulsión será un motor de corriente continua de rotor devanado, con conmutación por escobillas, y estator con imanes permanentes, CC (DCmotor). La razón de esta elección es por los requisitos de su función (algo exigentes) en cuanto a par y potencia, en un tamaño reducido, y a un coste bajo.

- En el caso del gobierno se usará un servo, (movimiento angular reducido, preciso, y par y velocidad reducidos).

No obstante, lo anterior y contrario a lo previsto al principio, hemos considerado una mejora necesaria incluir otro motor adicional para ajuste fino del trimado inicial de cada misión. Cualquier cambio en cantidad o posición de la carga útil del AUV requiere una corrección del estado de pesos, ya que cada misión debe iniciarse con flotabilidad nula y trimado cero. Ese motor adicional servirá para facilitar un trabajo iterativo que ejecutado manualmente sería tedioso. Nuestro objetivo es que pueda realizarse de modo automático, haciendo uso de los sensores de que dispone el AUV: ángulo de trimado, profundidad (control de presión) ....

Resumiendo, los elementos motrices previstos son 3:

- El motor (principal) del control de propulsión.

- El motor de maniobra (genera escora, puede ser un servo), así como,

- El husillo motorizado adicional para corregir el trimado fino inicial en cada misión.

Seguidamente vamos a describir algunas operaciones habituales relacionadas con la navegación del AUV. Más adelantes seguiremos con la descripción detallada de los motores y sensores, y finalmente describiremos la gestión de energía y la disposición de la planta.

6. La operativa de la navegación del AUV.

6.1 Descripción del proceso de ajuste de trimado y flotabilidad al inicio de la misma.

Hemos dicho que cada misión debe iniciarse con flotabilidad nula (neutra) y trimado cero. Cualquier cambio en cantidad o posición de la carga útil del AUV requiere una corrección del estado de pesos.

Olvidemos de momento el control de escora (asumamos que el AUV está adrizado y estable), y nos centramos en los otros 2 motores DC. En particular vamos a revisar el trabajo que hacen combinado sus acciones para el ajuste de trimado y flotabilidad para el inicio de la misión. ¿Qué hacen?, ¿Por qué interactúan?:

- La corrección del trimado inicial en cada misión consiste en mover un peso fijo dado, por medio de un mecanismo tuerca husillo, pero de baja potencia y velocidad, que se usará para adaptar la posición longitudinal del centro de gravedad de pesos a las condiciones hidrodinámicas óptimas del AUV en cada misión.

- También hará falta un mecanismo para ajuste fino de la flotabilidad. Y normalmente ese ajuste afecta al trimado (no ocurre así a la inversa), ya que la variación de flotabilidad está localizada en el émbolo, en proa.

Para los 2 ajustes requeridos (flotabilidad y trimado) usaremos 2 parámetros controlados:

- La posición inicial del husillo del émbolo (capaz de corregir el desplazamiento inicial en unos 3 cm de carrera = unos 60 cm^3 de desplazamiento, independientes de la regulación adicional necesaria para navegar arriba y abajo (NAA). Descontado ese desplazamiento (recorrido) inicial, y fijo para cada misión, el émbolo debe disponer

además de un recorrido continuo adelante y atrás, que asegure la flotabilidad requerida, positiva o negativa, en cada maniobra alternante de NAA.

- La posición del husillo del peso longitudinal regulable para trimado fino. Para este propósito podemos usar un motor de baja potencia y par, pero de regulación fina. Un stepper cualquiera de bajas prestaciones nos debería servir. Podemos elegir el tipo stepper sin mayor complicación porque tenemos una placa electrónica disponible para su uso sencillo, Y aprovechamos la oportunidad didáctica de su uso.

Con un pequeño esquema vamos a establecer cuáles son las relaciones entre los parámetros involucrados en la expresión del equilibrio de momentos trimantes:

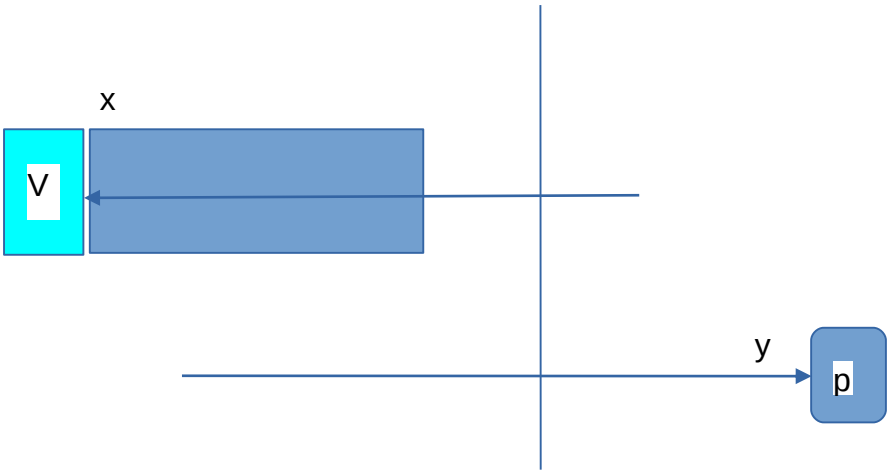


Ilustración 1 Esquema con la cámara de flotabilidad y el lastre móvil deslizante longitudinalmente. Fuente Propia

Este esquema representa varios elementos:

A la izquierda, el cilindro de cambio de flotabilidad, con el émbolo desplazado hacia popa (hacia la derecha en el dibujo) una cantidad, de modo que el volumen que el agua exterior ocupa dentro del cilindro, por la cara de proa del émbolo es V. El parámetro que determina la posición del émbolo es x,

creciente hacia la izquierda (hacia la proa).
En el mismo esquema, a la izquierda se representa un peso móvil, p, que permite modificar la posición longitudinal del centro de gravedad del AUV. El parámetro que determina su posición longitudinal es y, creciente hacia la derecha (hacia popa).

Las dos flechas horizontales del diagrama representan las posiciones x, y dentro de sus respectivos ejes de desplazamiento longitudinal.

La línea vertical a la mitad del diagrama es una referencia para las mediciones de x e y.

Designaremos los siguientes parámetros:

- X, posición longitudinal del émbolo de flotabilidad respeto a la línea de referencia (arbitraria)

- Y, posición longitudinal del husillo del contrapeso (peso móvil) respeto a la línea de referencia (arbitraria)

- P, peso móvil

- V, volumen inundado del cilindro del émbolo, desde el límite exterior

- A, área del émbolo

- M1, momento del volumen V respeto a la línea de referencia

- M2, momento del peso p respeto a la línea de referencia

- El volumen V se debe convertir en peso equivalente (de agua), para obtener magnitudes homogéneas. Podemos asumir densidad = 1000kg/m3

Si el AUV está totalmente sumergido, no existe área de flotación interceptada en la superficie del agua. En consecuencia, tanto el trimado, como la estabilidad transversal responden a una fórmula del tipo:

- M = DESPLA.GB. Sen(teta)

Donde:

- DESPLA, desplazamiento del AUV
- GB = KB – KG, parámetro de estabilidad
- KB altura del centro de carena
- KG altura del centro de gravedad
- M, momento resultante de las fuerzas de flotabilidad y peso
  Teta, ángulo de inclinación

La condición de equilibrio de momento trimante es:

- Mo + M1 + M2 = GB sen(trim)
- Mo, momento trimante de todos los pesos y empujes del AUV, salvo los relacionados con M1 y M2.

Si consideramos una variación elemental de algún parámetro con influencia en M1+M2, pero no en Mo y asumiendo GB constante, y trim pequeño para poder linealizar el seno, tendremos:

- dM1 + dM2 = GB. dTrim

Desarrollemos las relaciones que definen M1 y M2:

- V = A.d = A. (xf-x)

- M1=ro.V.(x+(xf-x)/2) = ro.V. (xf + x)/2 = ro.A.(xf-x).(xf+x)/2 = ro.A/2.(xf^2-x^2)

- M2 = p.y

- dV = -A.dx

- dM1 = ro.A/2.(-2.x.dx) = -ro.A.x.dx

- dM2 = p.dy

- d= xf – x,

- xf posición máxima, fija, límite admisible para x

Sustituyendo en la ec de trimado resulta:

- dM1 + dM2 = GB. dTrim

- -ro.A.x.dx + p.dy = GB. dTrim (***equilibrio de trimado***)

Esta última ecuación, que llamaremos *ecuación de equilibrio de trimado* es una de las que rigen nuestro modelo.

Otra ecuación que podemos establecer es la que refleja el ***equilibrio de fuerzas verticales***. Recordemos que la resultante de fuerzas de flotación y empuje va a suponer un movimiento vertical del AUV. En una primera aproximación puede admitirse una relación lineal entre la velocidad vertical, y la resistencia hidrodinámica de fricción que se opone a esa velocidad. La resultante de todas las fuerzas verticales es, según el principio de Newton, será el producto masa. aceleración vertical:

- Peso – empuje = k. vel_vertical + peso/ro. d(vel_vertical)/dt

- Peso – (Emp inicial - ro.V) = k. vel_vertical + peso/ro. d(vel_vertical)/dt

- (Peso – Emp inicial) + ro.V = k. vel_vertical + peso/ro. d(vel_vertical)/dt

Si derivamos d/dt, resulta otra ecuación en la que podemos imponer que la variación de la aceleración también sea nula, lo que significa que es un punto de equilibrio estable.

- 0 + ro.dV/dt = k. d(vel_vertical)/dt + 0

- 0 + ro.dV/dt = k. d(dz_{cdg}/dt)/dt

- ro.A. dx/dt = k. (zpp_punto2 – L/2.trim_punto2),

- ro.A. x_punto = k. (zpp_punto2 – L/2.trim_punto2)

Esta última ecuación, por si sola, relaciona las oscilaciones verticales y la variación de la posición del émbolo. Si el émbolo está quieto, la condición se reduce a que las aceleraciones a las que está sometido el extremo de popa son únicamente las debidas a la oscilación de trimado.
zpp_punto2 – L/2.trim_punto2

Siendo:

- Peso, el peso total del AUV, para la misión. Este peso es *constante durante toda la misión*, salvo inundación o pérdida de elementos externos (lastre fijo, etc.)

- Empuje, el volumen de agua desplazado en un instante dado, multiplicado por la densidad del líquido.

- Empuje inicial, el volumen de agua desplazado por el AUV, con el

émbolo de flotabilidad en la situación de máxima flotabilidad (totalmente a proa). Es un *valor fijo durante toda la misión*, salvo inundación o pérdida de elementos externos.

- V, volumen del cilindro de flotabilidad que queda inundado por el agua del exterior, a medida que el émbolo se desplaza hacia popa

- V el vertical, es la velocidad vertical a la que se hunde el AUV. Se asume positiva hacia profundidad creciente.

- V el vertical = dz_{cdg}/dt.

- Z, profundidad del punto de referencia de medida del AUV.

    - Si el sensor de profundidad no está en el centro del AUV, sino en un extremo, por ejemplo en popa, la conversión será:

    - Zpp =Zcdg +L/2.trim,

        - L, longitud del AUV

        - Trim, angulo de trimado, Medido en radianes, y sustituyendo a su tangente trigonométrica linealizada. Se considera positivo si es apopante (popa mas sumergida que la proa).

        - Zcdg, Zpp, profundidad en los puntos de referencia cdg, pp

    - Zcdg = Zpp – L/2.trim

- K, factor de proporcionalidad que contempla la resistencia hidrodinámica al avance en sentido vertical del AUV en la condición de trimado trim, a la velocidad de descenso vel_vertical. Es desconocido,

pero es constante para esas condiciones.

.

Operando en la última ecuación podemos escribir:

- (peso – Emp inicial) + ro.V = k. vel_vertical
- (peso – Emp inicial) + ro.A. (xf – x) = k. dif(zpp – L/2.trim) /dt

Deseamos que la velocidad vertical sea nula, lo que significa:

- Ro. V = (Emp inicial – peso)

Es decir, el volumen V (que queda inundado) en el cilindro de flotabilidad deberá compensar la diferencia entre el empuje y el peso en la condición de referencia de inicio de misión. Eso ya lo sabíamos. Bastaría con calcular o medir empuje y peso para estimar V. Sin embargo, no queremos tener que medir y calcular. En su lugar esperamos que los sensores del AUV detecten el movimiento y encuentren el punto de equilibrio, en el que el movimiento tiende a anularse.

Si derivamos respecto al tiempo la ecuación de fuerzas verticales tenemos:

- ro. dV/dt = k. d(vel_vertical)/dt
- ro. dV/dt = k. d(dif(zpp – L/2.trim) /dt)/dt
- ro.A.dx/dt = k. d(dif(zpp – L/2.trim) /dt)/dt
- ro.A.x_punto = k. (zpp_punto2 – L/2.trim_punto2)
- ro.A/k. x_punto = (zpp_punto2 – L/2.trim_punto2)

y además sea una condición de equilibrio estable, por lo que su primera diferencial también deberá se nula. Si calculamos variaciones infinitesimales:

- 0 – ro.dV = k. dvel_vertical

Dividiendo miembro a miembro eliminamos el factor k, y resulta:

- ro.dV/(Po – ro.V) = dvel_vertical/vel_vertical

HASTA Aquí las ecuaciones están claras. En lo sucesivo hay que gestionar como resolver para encontrar la solución:

- X, y, V, trim=0, velvert=0

6.2 Determinación de los parámetros iniciales de la misión. Rutina de ajuste automático o auto-calibrado.

El siguiente esquema ayuda a entender el proceso:

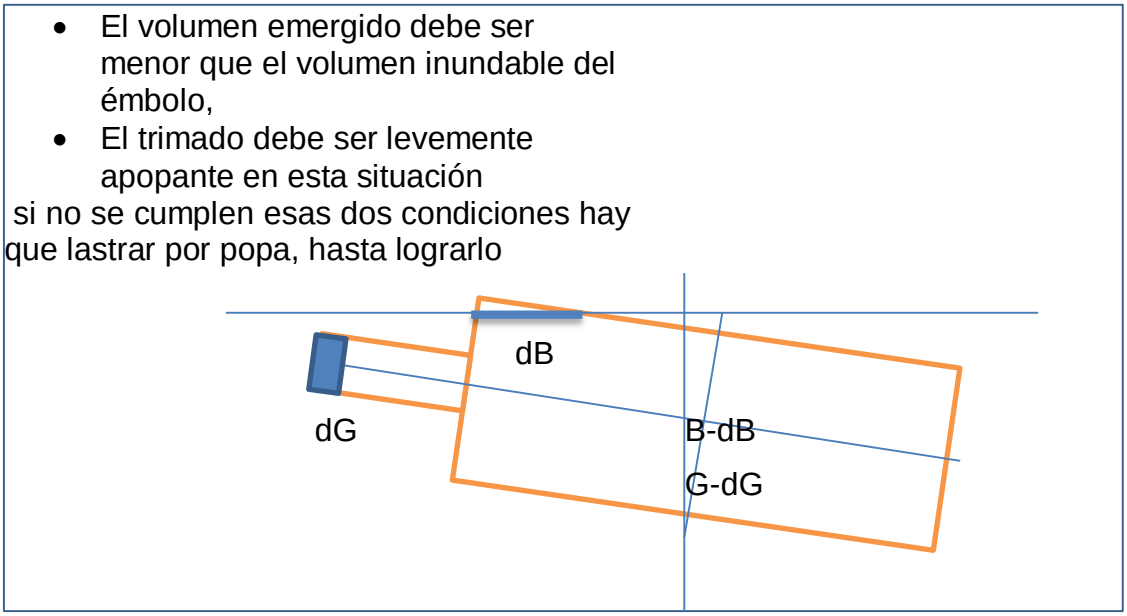


Ilustración 2 Esquema del AUV en reposo en la superficie. Fuente Propia.

LA FORMA DE OPERAR ES:

1. Se mete el AUV en el agua, con el émbolo posicionado al límite de su recorrido hacia proa (condición de desplazamiento de carena máximo). En esa condición el AUV debería FLOTAR.

   A. Si NO FLOTA, hay que quitar algo del lastre fijo, y volver a la posición de flotabilidad positiva inicial.

   B. Si FLOTA, se continua así:

      i. NO sumerge el borde inferior frontal del cilindro de flotabilidad. Tiene flotabilidad EXCESIVA, y hay que aumentar la cantidad y/o posición del lastre fijo hasta que se logre.

ii. SI sumerge el borde inferior frontal del cilindro de flotabilidad. Se puede iniciar la secuencia de inundación.

1. Se inicia el movimiento del émbolo, x, hacia popa, haciendo que se inunde la parte externa del cilindro de flotabilidad, hasta que se alcanza un punto xo en el que empiece a hundirse todo el AUV.

    a. Si se alcanza ese punto de HUNDIMIENTO, la distancia xfinal – xo es la diferencia de flotabilidad inicial, y se puede continuar con la *secuencia de medida automática de parámetros*.

    b. Si NO se alcanza ese punto de HUNDIMIENTO, habrá que añadir algo de lastre fijo. Se hace y se vuelve a la posición de flotabilidad positiva inicial. El volumen de carena no sumergido, dB en la figura siguiente, debe ser del orden de la mitad del volumen inundable del cilindro del émbolo.

2. Una vez que el AUV flota, se inicia la secuencia de medida automática de parámetros, en la que se van midiendo en periodos ctes los stes valores:

    a. x, z, trim

    b. Con ellos se calculan otros:

        i. x_punto
        ii. z_punto
        iii. z_punto2
        iv. trim_punto
        v. trim_punto2

    c. con el vector de medidas, unas 10, por ejemplo, se tiene información

para calcular hasta otros 10 valores de k, con los que se puede determinar su precisió.

3. De lo anterior se puede establecer el valor de:

    a. k, resistencia dinámica viscosa al hundimiento.

    b. El valor de xo a partir del cual empieza el hundimiento, con lo que queda establecido V inicial de la misión

4. En este momento el AUV ha descendido hasta zmax prueba, y/o han pasado unos segundos, periodo prueba.

5. A continuación, se debe poner a trimado =0, y flotabilidad neutra a cota z = zmin = 0.05 m = 5 cm de agua sobre la generatriz del cilindro mayor de AUV. para ello se da la consigna calculada:

    a. x= Xo

    b. y = yo (posición longitudinal del contrapeso p, para regular el timado inicial)

6. Eso debe llevar a una posición
    a. Zo
    b. trim0=0
    c. Si no es así se debe modificar con un pequeño movimiento en x, y

7. Finalmente, en un periodo de tiempo se alcanza la posición, con error menor que la tolerancia deseada.

8. Durante el proceso de hundimiento, una vez iniciado, se puede calcular GB a partir del mov del peso escorante, y deducir el ratio mov servo, escora AUV, y de ahí sacar el parámetro GB, de estabilidad.

CON TODO lo anterior quedan definidos los finales de carrera operativos del émbolo. No son finales hardware sino software.

Los finales hardware si son físicos y corresponden a los extremos admisibles de la carrera del émbolo. Los finales software son definidos por variables en memoria y quedan delimitados por el recorrido útil que se puede hacer dentro del émbolo, para que se pueda realizar una inmersión a máxima velocidad, y una emersión correspondiente también a la misma máxima velocidad. Cualquier maniobra a menor velocidad puede resultar problemática, pues la sustentación de las alas y escora pueden ser insuficientes.

Para maximizar el potencial del AUV hay que conseguir que Xo, punto de flotabilidad neutra, este situado en el punto medio entre las posiciones extremas admisibles para el émbolo, entre xf (limite por proa) y xff (límite a popa).

La diferencia que pueda existir se puede corregir con variaciones de lastre fijo.

Finalizada la etapa de calibrado, ya se tienen los parámetros que van a permitir el control de la navegación durante la misión. Se archivan en algún modo de memoria no volátil, y quedan listos para posteriores misiones similares.

6.3 Ejemplo de la rutina de navegación alternante, o navegación arriba y abajo (NAA):

Partiendo de los parámetros conocidos de navegación, y suponiendo que el AUV se encuentra en posición relativamente horizontal (poco trimado) cerca de la superficie, la secuencia es:

1. Establecer los parametros iniciales de misión, zmin, zmax , tmax.
2. Modificar x, reduciendo, para aumentar el volumen inundado, hasta la consigna deseada de z_punto = z_punto_descenso.
    a. Ir trazando los valores durante ese recorrido:
        i. x
        ii. z
        iii. trim
    b. Calcular.
        i. x_punto
        ii. z_punto
        iii. trim_punto
        iv. z_punto2
        v. trim_punto2
        vi. monitorizar los valores y determinar los puntos característicos de la trayectoria.:
            1. z_punto y el potencial de aceleración z_punto2,
            2. Si se alcanza z_punto2 =0, ya se ha alcanzado la máxima velocidad de descenso
            3. Asumiendo que el ángulo de trimado y el de trayectoria difieren en unos 2º, se puede determinar la traza de esa curva, trayectoria_se puede iniciar una rutina de aceleración jugando con el trimado según x.
        vii. Se puede empezar a planificar la inversión del movimiento, para ascender.
            1. Se debe estimar el tiempo necesario para alcanzar zmax, conocer/estimar la velocidad axial mínima, así como su velocidad proyectada sobre el plano horizontal
            2. Hay que lograr que el paso por zmax sea lo más rápido posible, y con trima =0. Se trata de que no se pare, pues en ese momento también se pierde la maniobrabilidad, control de rumbo, etc.
            3. Después de pasar por trim=0, y/o invertir el signo de z_punto, estaremos en fase ascendente, y se

monitoriza como en el caso descendente ya descrito.

4. Se continúan las alternancias arriba y abajo hasta alcanzar tmax, tiempo máximo de la misión.
5. Se pasa al estado de subir a la superficie para recogida.
6. En el estado de permanencia en la superficie para recogida sería adecuado que se encendiera un led (en modo estroboscópico, para aumentar su visibilidad) situado en el extremo superior de proa del cilindro principal de AUV, dentro del casco. Eso puede facilitar la detección del AUV.

6.4 Control de cambio de rumbo.

El rumbo del AUV se determina a partir de su actitud, calculada con los datos aportados por el sensor inercial IMU. Eso nos da una dirección (en grados, relacionados con el norte magnético) del eje longitudinal (proyectado sobre plano horizontal) del AUV.

Como ocurre con los planeadores, el cambio de rumbo en el AUV lo logramos generando una escora. En nuestro caso hacemos un desplazamiento lateral de un peso, que nos dá una escora. El AUV iniciará un giro con centro en el lado del ala baja, cuando el AUV desciende, o con centro en el lado del ala alta cuando el AUV asciende. A mayor escora menor radio de giro.

El desplazamiento lateral de peso lo realizaremos con un servo que hace girar el peso escorante dentro de un plano perpendicular al eje longitudinal del AUV.

6.5 Navegación en espiral o tirabuzón.

Una modalidad especial de navegación, con utilidad en pruebas, puede ser la navegación describiendo espirales ascendentes y descendentes, de modo que el AUV esté siempre dentro de una columna de agua de un diámetro limitado. Esto debería facilitar la localización visual en cualquier momento, así como navegar en zonas restringidas.

Hay que tener en cuenta que si la trayectoria espiral es siempre dejando el centro de giro a la misma banda, habrá que modificar la caída de las alas (escora) para 'planear' en coherencia con esa espiral. Así, al bajar, el centro de giro estará en el lado del ala más baja. Mientras que al subir las posiciones serán 'a espejo', sobre un plano horizontal, de las anteriores.

6.6 Flotabilidad en emergencia. Rescate.

La condición extrema de un fallo de seguridad sería la pérdida de energía, o una pérdida de gobierno que lleve al AUV a una maniobra suicida (inmersión más allá de la profundidad máxima admisible). También podría ser una inundación, o un fallo estructural. En cualquiera de esas situaciones extremas, la respuesta ha de ser intentar volver a la superficie.

Un colapso estructural o una inundación masiva solo podrían ser contrarrestada con un diseño del AUV compartimentado en zonas independientes, de modo que el fallo en alguna de ellas no suponga el fallo de las demás. Así ocurre en los sumergibles 'reales' con responsabilidad. No obstante, la complejidad estructural que eso supondría para nuestro caso nos lleva a renunciar directamente a esa seguridad. Nos queda centrarnos en los otros aspectos de la seguridad

Si se produce una pérdida en la lógica o en la energía, el sistema de gestión debe pasar automáticamente a vuelta a casa (con la energía disponible), o si todo empeora (por pérdida total de energía), el sistema pasa a flotabilidad positiva, modo rescate

Hay que hacer un cuadro de lógica con las diversas opciones

- Fallo en una fuente (activa o en STBY) -> pase a modo Vuelta a casa. Poner émbolo en posición de ascenso, con las fuentes disponibles, sin necesidad de lógica, solo fin de carrera.

- Fallo en las 2 fuentes. (Antes de llegar aquí debería pasar por el punto anterior, pero…) significaría pérdida total del control. Se inicia el modo rescate. Se debería pasar a una flotabilidad de emergencia, por suelta pasiva de un lastre externo (posiblemente en la ojiva de popa). Si se suelta un lastre de la ojiva de popa, el AUV tiende a subir la popa e iniciar un movimiento opuesto al normal, lo que facilitaría salir de un

encallamiento accidental producido navegando avante.

Además. En el modo de rescate (después de pérdida de total de energía), el sistema debe pasar automáticamente, y sin necesidad de energía, a un estado de flotabilidad positiva, que lleve al AUV a la superficie. El procedimiento usual sería que la falta de energía activa el disparo de un gatillo que libera (suelta) un bloque externo de lastre sólido. Adicionalmente, es deseable que también ese lastre sólido pueda ser rescatado posteriormente. Para ello, el bloque de lastre deberá tener adujado un cabo con un chicote ligado al lastre, y el otro a un flotador, o al propio AUV. La aduja de ese cabo debe ser tal que nunca se enrede cuando se libere. El extremo unido al flotador (o al AUV) subirá a la superficie, y será visible y recuperable. A partir de ahí debería ser posible recuperar del fondo el citado lastre sólido.

El cabo podría estar ligado a un flotador permanente (pelota de ping-pong no soluble en agua, y resistente a la presión operativa del AUV), que se desenrolle cuando el dispositivo de corte quede liberado por la pérdida de energía. La capacidad de ese flotador será simplemente la necesaria para forzar el deshacer (sin enredos) las adujas del cabo. Hasta su total extensión. Si el cabo tiene además flotabilidad neutra y es adecuadamente flexible se mejora el proceso.

Por su parte, el AUV, con flotabilidad positiva subirá hasta la superficie y debería quedar visible a flor de agua. Quizá con un led estroboscópico en la zona de proa, la más emergente.

Otra cuestión a valorar es determinar cómo ha de ser disparado el sistema de rescate. Las alternativas que se ven son:

1. El disparo puede hacerse forzado o automático. El modo forzado permitirá hacer pruebas. El modo automático se dispararía así: Cuando la tensión en el punto de control es inferior a Vmin, se debe producir la alarma y la activación (cierra circuito) de un circuito eléctrico independiente con una pila independiente, que

active un electroimán que tire del gatillo de disparo, que (con el juego de palancas debido) libere el muelle de descarga del lastre. Esta opción es algo complicada, aunque no genera consumo de corriente durante el tiempo en que está en espera. Por contra, requiere mantener acumulada energía en una reserva, para poder dar el “último suspiro”. Si esa reserva fallase, no se dispararía.

2. La otra alternativa es que un electroimán esté permanentemente activado, reteniendo un muelle antagonista. Esta opción asegura que en algún momento se dispara, aunque tiene el inconveniente de que el consumo continuo del electroimán acorta la duración de la reserva de energía. Cuando falle la corriente de alimentación del electroimán, el muelle antagonista forzará llevar al gatillo a la posición de disparo mecánico.

3. Otra opción intermedia es que exista una palanca basculante, con un flotador en el extremo, que lo levante, estando dentro del agua (por flotabilidad), mientras que fuera del agua, por su peso tiende a estar bajo. Además de lo anterior, un electroimán mantendrá retenido la palanca del flotador en posición baja, solo si el electroimán está activo, y eso solo ocurrirá si hay un elemento sensible al agua que detecte que está dentro del agua, y no fuera. Esto hace que el periodo de consumo del electroimán se reduzca sólo a los periodos en que esté dentro del agua, no fuera.

4. Otra variante podría ser, como en las balsas salvavidas, un sistema de liberación automática activado por presión. Pero esto no subiría al AUV si este queda caído, posado en el fondo a una profundidad menor que la de disparo. Es decir, no garantiza que el AUV suba a la superficie.

5. La única buena parece la primera.

En cualquiera de las opciones, el mecanismo para el disparo puede inspirarse en el de las pistolas, nuez de ballestas, disparadores de arco, palancas encadenadas, etc. y hacer las piezas finamente mecanizadas en 3Dprinter. de modo que una energía eléctrica moderada pueda vencer un muelle antagonista, y la resistencia del sistema, y con ello deslizar el gatillo hasta el disparo. Este circuito funciona en el modo Normalmente no energizado, pero su fuente ha de estar continuamente monitorizada frente a cambios de tensión o bajada de tensión.

7. Gestión de los motores.

Deseamos operar con el Arduino varios motores, de tipos diversos, y queremos conseguir un buen manejo. Es decir, controlar posición, velocidad, consumo, etc. De modo que podamos tomar decisiones automáticas a partir de esa información.

Dentro del entorno Arduino, el estándar para manejar motores con comodidad es usar una combinación de hardware y software.

El hardware es una placa MotorShield, que está orientada a la alimentación con potencia controlada y regulada de motores DC, servos y Steppers.

El software es un conjunto de librerías orientadas a algunas aplicaciones.

7.1 Caso de Servos y steppers.

Los servos y steppers tienen unas características orientadas a un manejo preciso, fiable y fácil de su posición en cada momento, por lo que son habituales en las aplicaciones electromecánicas. Llevan incorporados elementos de detección de posición y son capaces de adaptar su alimentación según la consigna de posición recibida.

Los servos reciben una consigna de posición angular, y modulan su consumo para aplicar de modo inmediato el par necesario para alcanzar esa posición.
Los steppers o motores paso a paso, avanzan cantidades discretas iguales, siguiendo una consigna, y consumiendo para ello la intensidad necesaria para vencer en cada instante el par resistente.

7.2 Caso de motores DC con encoder.

El caso de los motores DC es algo diferente. Por si mismo carece de los elementos de control, por lo que hay que añadírselos.

Para poder mantener con precisión una velocidad (aunque el par resistente sea variable) es necesario cerrar el lazo de control con una realimentación de información de la velocidad, y con ella determinar la variación de tensión de alimentación que se requiere para lograr aproximarnos a la velocidad de referencia que haya sido consignada.

La velocidad de giro del motor en cada momento la podemos conocer a través del encoder, que crea unas señales que deben ser interpretadas adecuadamente.

La alimentación en tensión puede ser modulada (vía PWM, por ejemplo) para adaptarla a lo que sea requerido de modo inmediato.

Solución general adoptada para el conjunto de los diversos tipos de motor
La generalización de esas funcionalidades, tanto para los motores DC con encoder, como para los servos y steppers, la vamos a lograr encadenando varios ‘módulos’.

Driver de motor, normalmente un puente H, que ‘conmuta’ la tensión aplicada a los terminales de potencia del motor. Ese conjunto de hardware se suele agrupar en una placa que llamaremos MotorShield, que será nuestra opción.
Encoder, para medir velocidad y sentido de giro del motor. Eso también nos va a permitir conocer la posición angular del rotor en cualquier momento. Se gestiona con una librería específica para Arduino.

Control o regulación, a través de un PID., control Proporcional, Integral y Diferencial. Necesitaremos una librería de Arduino para operarlo con comodidad.

7.3 La placa MotorShield.

Su principal función es separar los circuitos eléctricos de control, (baja potencia), de aquellos de fuerza (mayor potencia), para la gestión de motores eléctricos varios.

Para la potencia solemos usar una fuente de alimentación adecuada en tensión e intensidad para el consumo esperable.

Para el control usamos tensiones e intensidades pequeñas. En los estándares TTL, 5 V, 20 mA.

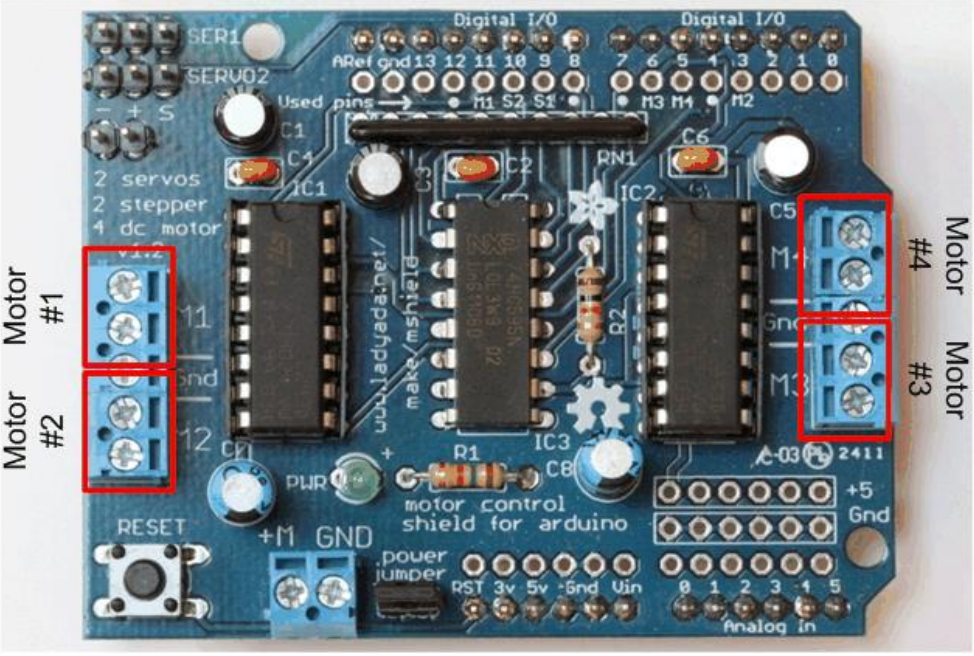


Ilustración 3 Placa MotorShield. AdaFruit MotorShield.

7.3.1 Pines usados y libres.

La gestión de motores la haremos a través de una MotorShield, como la de nuestro interés (AdaFruit AFMotorShield v1.0, ahora obsoleta frente a las más modernas V2). La motorShield ocupa algunos pines del Arduino, y otros quedan libres y disponibles para otros usos. Dentro de los libres tenemos que

seleccionar algunos para dedicarlos a la gestión de los encoders, y a otras funcionalidades de nuestro AUV, o de otras Shields. (Data logger, etc…)

Esto va a dar lugar a tener que hacer una tabla de mapeo de sistemas y asignación de pines.

La información de la AFMotorShield está aquí:

https://www.adafruit.com/product/81

En particular hay una buena descripción en pdf, sacada de esa web, que es

https://cdn-learn.adafruit.com/downloads/pdf/adafruit-motor-shield.pdf

En la pag 7/58 de ese pdf se indica el uso de los pines.

What pins are not used on the motor shield? All 6 analog input pins are available. They can also be used as digital pins (pins #14 thru19)

Digital pin 2, and 13 are not used.

The following pins are in use only if the DC/Stepper noted is in use:

Digital pin 11: DC Motor #1 / Stepper #1 (activation/speed control)
Digital pin 3: DC Motor #2 / Stepper #1 (activation/speed control)
Digital pin 5: DC Motor #3 / Stepper #2 (activation/speed control)
Digital pin 6: DC Motor #4 / Stepper #2 (activation/speed control)

The following pins are in use if any DC/steppers are used

Digital pin 4, 7, 8 and 12 are used to drive the DC/Stepper motors via the 74HC595 serial-to parallel latch

The following pins are used only if that particular servo is in use:

Digitals pin 9: Servo #1 control

Digital pin 10: Servo #2 control

Which pins are connected to the DC/Stepper motors?

The DC/Stepper motors are NOT connected to the Arduino directly. They are connected to the 74HC595 latch which is spoken to by the Arduino. You

CANNOT talk directly to the motors, you MUST use the motor shield library.

Huh? I don't understand...

You can try reading this nice overview written by Michael K (http://adafru.it/aO9)

How can I connect to the unused pins?

The analog pins (analog 0-5 also known as digital pins 14-19) are broken out in the bottom right corner.

Pin 2 has a small breakout since its the only truly unused pin

The remaining pins are not broken out because they could be used by the motor shield. If you are sure that you are not using those pins then you can connect to them by using stacking headers when assembling the kit or soldering onto the top of the header with wires, or using a "Wing shield"

Según lo anterior: L

Las posiciones marcadas dentro de rectángulos verdes en las imágenes siguientes son pines que pueden estar libres, según los motores conectados. Tales pines son lo que podremos utilizar para otros fines

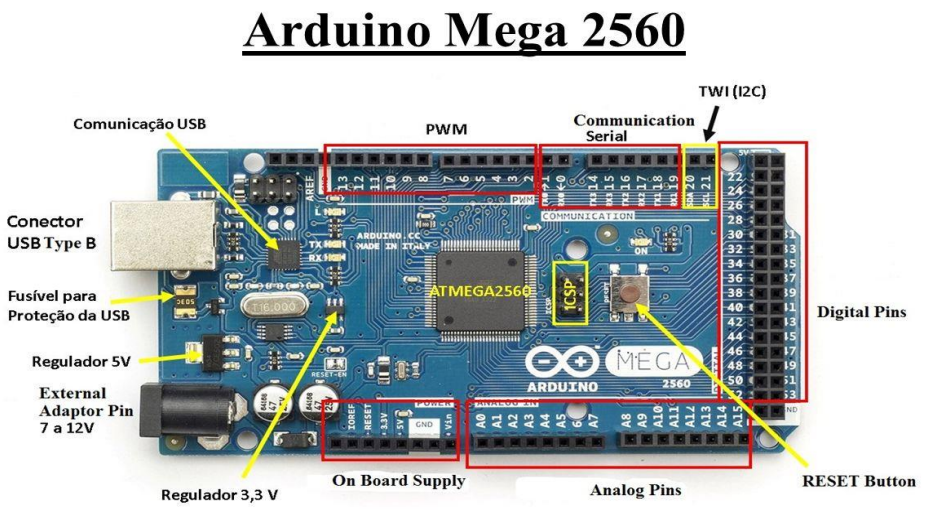


Ilustración 4 Arduino Mega 2560 AndaFruit MotorShield

Si disponemos solo un DCmotor en el terminal #1, los pines usados en la Mega serán: 4,7,8,12,11
Podemos añadir otro DCmotor en terminal #2, usando otro pin más: 3

Podremos añadir servos en Pin: 9, 10

Y todos los demás estarán libres (aunque físicamente no sea fácil establecer un contacto eléctrico seguro).

Para vencer la dificultad de establecer contactos eléctricos en los pines libres (no usados por la shield), se usan en ocasiones regletas de conectores que son macho por debajo y hembra por encima, y que permiten apilar shields en paralelo. Otra posibilidad es crear una mini gaza en el terminal del cable eléctrico, y hacer que el pin la atraviese, antes de que el shield quede enchufado en su posición. Esta solución es aceptable para el caso de necesitar rescatar unos pocos pines.

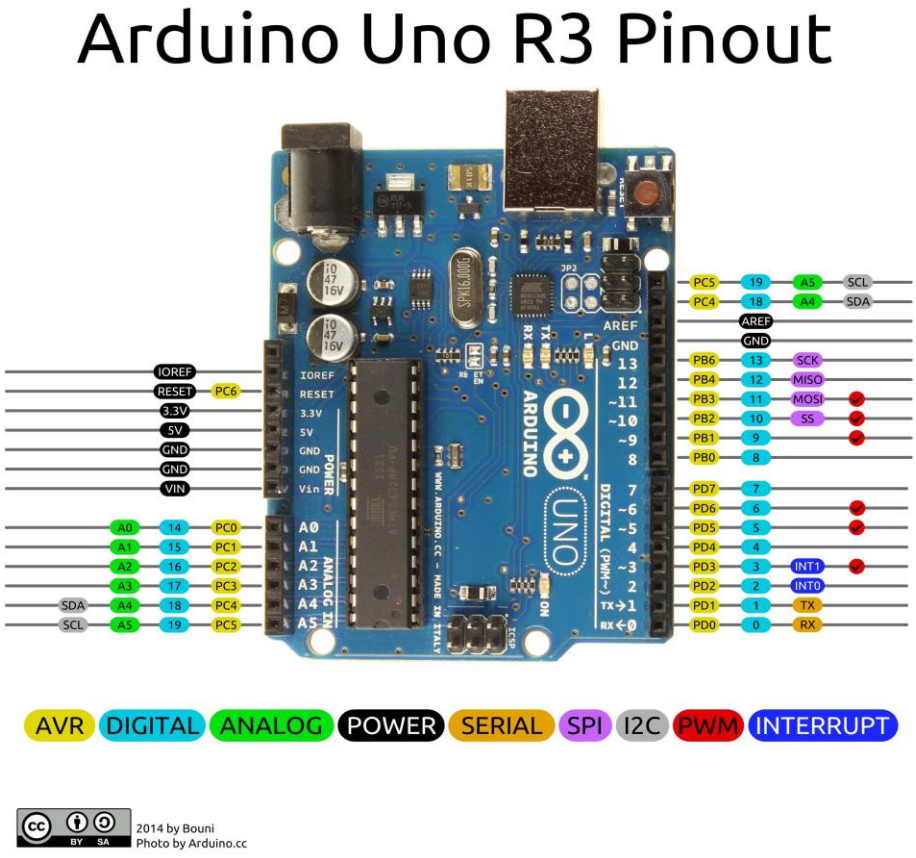


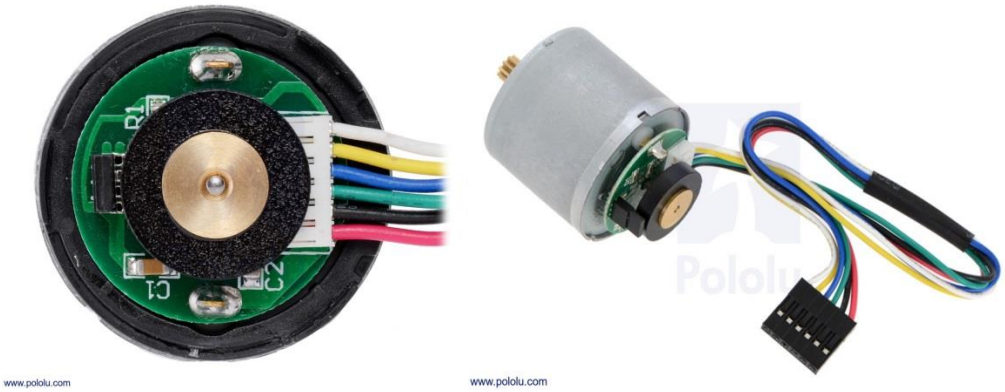
Ilustración 5 Arduino Uno R3 Pinout. AndaFruit MotorShield

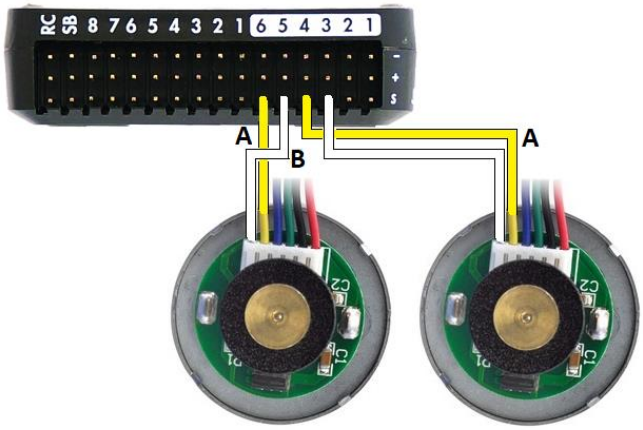
7.3.2El Encoder asociado al motor DC

El encoder más habitual en los motores DC que estamos manejando es el descrito en esta web:

https://www.pololu.com/product/2821

Se trata de uno de tipo rotativo incremental, en cuadratura, magnético de efecto hall, de dos canales, con una resolución total máxima de 64 CPR (counts per revolution). Es coaxial con el motor, y suele disponerse en la parte trasera, lado opuesto al extremo frontal eje motor





Color	Function
Red	motor power (connects to one motor terminal)
Black	motor power (connects to the other motor terminal)
Green	encoder GND
Blue	encoder Vcc (3.5 – 20 V)
Yellow	encoder A output
White	encoder B output

Ilustración 6 Motor DC y cableado. Pololu.

How to read magnetic incremental encoder specs - yc-52010:
https://www.cnczone.com/forums/servo-motors-drives/352768-cnc.html

YC-52010 series permanent magnet DC brush 550 motor AB phase bidirectional Hall magnetic encoder:

https://www.yoycart.com/Product/42299022187/

Otras referencias:

Una descripción completa sobre transductores de movimiento (en inglés)

http://irtfweb.ifa.hawaii.edu/~tcs3/tcs3/0306_conceptual_design/Docs/05_En

coders/encoder_primer.pdf

Una librería muy perfecta y adecuada para Arduino UNO, y MEGA (), que gestiona los 2 pines requeridos para el encoder de cada DCmotor.

https://www.pjrc.com/teensy/td_libs_Encoder.html

Author	Paul Stoffregen
Website	http://www.pjrc.com/teensy/td_libs_Encoder.html
Github	https://github.com/PaulStoffregen/Encoder
Category	Signal Input/Output

Los pines que seleccionemos para el tráfico de cada encoder deben ser, además, compatibles con los demás usos y Shields del AUV y del Arduino.
En el apartado pines usado y libres, de la MotorShield, vimos lo pines que se utilizan (internamente por las librerías y el shield). Los demás, podemos usarlos para otras tareas en nuestra programación.

En el caso de los encoders, la forma habitual en que son gestionados es mediante el uso de las interrupciones del procesador. El Arduino tiene una gestión de interrupciones que va a permitir ese uso.

8. Interrupciones del procesador

Las interrupciones permiten que el procesador pueda gestionar sucesos esporádicos sin apenas consumir recursos de tiempo de proceso, que así pueden destinarse a otras tareas mientras no se produzca la interrupción.

Gran parte de los eventos relacionados con la seguridad de nuestro AUV los vamos a gestionar por medio de interrupciones. Podremos establecer varios tipos de interrupciones, y cada uno de ellos dará lugar a un procedimiento diferente. Finalmente hay que establecer una tabla resumen de tales eventos, que será esencial en el diseño del sistema de navegación.

Algunas referencias didácticas sobre interrupciones:

https://www.luisllamas.es/que-son-y-como-usar-interrupciones-en-arduino/

https://www.prometec.net/interrupciones/

https://programarfacil.com/blog/arduino-blog/interrupciones-con-arduino-ejemplo-practico/

Uno de mis DCmotores con encoder:

https://es.aliexpress.com/item/6V-100-210-300RPM-Encoder-Motor-DC-Gear-Motor/32826765244.html

Queda claro que el encoder de este es de 11 pasos por vuelta del DCmotor, sin reductora

La siguiente web es un ejemplo de este tipo de motores, y el código que lo controla, así como las conexiones que se requieren en varios tipos de Arduino: UNO, MEGA, …
https://www.dfrobot.com/wiki/index.php/12V_DC_Motor_251rpm_w/Encoder

_(SKU:_FIT0186)

https://hackaday.io/project/11382-dc-motor-speed-control-with-pid

El siguiente es parecido, pero incluye un receptor BT HC-05, para comandar con un móvil, pero los motores no tienen encoder.

https://www.hackster.io/Yogeshmodi/bluetooth-controlled-car-e8c90e
https://www.instructables.com/id/Arduino-Bluetooth-RC-Car-Android-Controlled/

8.1 Uso de LIBRERIAS adecuadas para Arduino, con PID, Kalman, etc.

Hemos dicho que nos apoyaríamos en librerías públicas (open source) para manejar ciertas operaciones:

- Control PID
- Gestión de la placa Motorshield.
    - Gestión de servos
    - Gestión de motores DC
    - (Nótese que la placa MotorShield no gestiona los encoders de los motores DC, por lo que se requiere software y hardware adicional)
- Gestión del Encoder en los motores DC.
- Gestión de una placa data logger, de registro de datos en tiempo real, (RTC).
    - https://cdn-learn.adafruit.com/downloads/pdf/adafruit-data-logger-shield.pdf
    - La que yo tengo es Adafruit, y parece ser la version revB, con el RTC PCF8523, compatibleR3 (arduino)
    - Los pines no usados por esta placa son casi todos ya que usa el 2x3 pins ICSP header, que libera de usar librerias especiales y otros pines para las comunicaciones I2C or SPI, (ver pag 22/70 del pdf)

- Etc...

Existe una librería PID bien documentada, de uso extendido:

http://brettbeauregard.com/blog/wp-content/uploads/2012/07/Guía-de-uso-PID-para-Arduino.pdf

https://www.lawebdelprogramador.com/cursos/Arduino/9007-Arduino-PID-Guia-de-uso-de-la-biblioteca.html

La versión 1.2 de 2017 de la librería de PID usual

https://github.com/br3ttb/Arduino-PID-Library

La lib PID mas popular del entorno arduino

http://playground.arduino.cc/Code/PIDLibrary

Otra lib complementariade la anterior para autotune los 3 parametros del PID

https://playground.arduino.cc/Code/PIDAutotuneLibrary

Control PID de posición con librería para Arduino. Un ejemplo de buena calidad MMBB sobre ello es el siguiente:

https://sites.google.com/site/proyectosroboticos/control-de-motores/control-pid-con-libreria

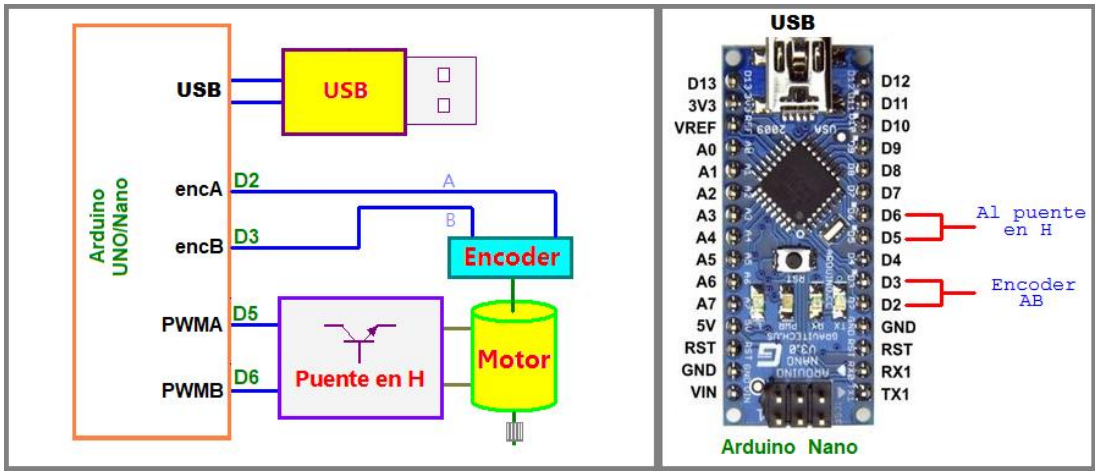


Ilustración 7 Esquema de control de motor con Arduino PID. Proyectos robóticos.

Este siguiente, relacionado con el anterior, es una mejora:

https://sites.google.com/site/proyectosroboticos/control-de-motores/control-pid-mejorado

Sobre el motorShiel , MotorDC y su librería:

https://learn.adafruit.com/adafruit-motor-shield/using-dc-motors

https://learn.adafruit.com/adafruit-motor-shield/af-dcmotor-class

https://cdn-learn.adafruit.com/downloads/pdf/afmotor-library-reference.pdf

Librerías sobre ENCODER

https://www.arduinolibraries.info/libraries/encoder

http://andrewjkramer.net/motor-encoders-arduino/

Con código y control de intensidad y sobrecarga, estado de batería, etc.

Un ejemplo que incluye PID, kalman, encoders, etc… Para un balancing robot.

http://forum.arduino.cc/index.php?topic=8652.0

https://forum.arduino.cc/index.php?topic=8652.0

https://forum.arduino.cc/index.php?action=printpage;topic=8652.0;images

Incluye explicaciones muy detalladas dentro del code. Son varias páginas. de ese último se puede sacar todo: PID, encoder, Kalman, suavizado de ruidos, etc.

Lo siguiente es la segunda parte de la anterior web, pero todo visto a la vez en una sola página.

http://forum.arduino.cc/index.php?action=printpage;topic=8871.0;images

Librerías para suavizado de datos –filtrado.

Una función importante es conseguir valores 'suavizado' a partir de sensores 'ruidosos'. Eso se consigue con diversos medios:

Filtro de Kalman, KF, UKF,

- https://github.com/alexvargasbenamburg/arduino_FiltroKalman
- https://github.com/micuat/ofxUkf
- https://github.com/Efreeto/UKF
- http://micorobot.blogspot.com/2012/11/filtro-de-kalman-e-imu.html
- http://www.starlino.com/imu_kalman_arduino.html
- http://bibing.us.es/proyectos/abreproy/91560/fichero/TFG+-+M%C2%AA+Esther+Aranda+Romasanta.pdf
- https://www.arduinolibraries.info/libraries/kalman-filter-library

Filtro complementario, y quaternions https://www.luisllamas.es/medir-la-inclinacion-imu-arduino-filtro-complementario/

Filtro de fusión de dato.

AHRS filter (madgwick, 2009), attitude and heading reference system

- https://github.com/kriswiner/MPU6050/wiki/Affordable-9-DoF-Sensor-Fusion
- http://x-io.co.uk/open-source-imu-and-ahrs-algorithms/
- http://x-io.co.uk/res/doc/madgwick_internal_report.pdf
- https://flyingcarsandstuff.com/2018/04/quick-easy-attitude-and-heading-reference-system/
- https://github.com/psychogenic/MPU9250/blob/MadgwickAHRS/examples/AHRS_SPI/AHRS_SPI.ino
- https://github.com/psychogenic/MPU9250/tree/MadgwickAHRS
- https://github.com/kriswiner/MPU6050/wiki/Simple-and-Effective-Magnetometer-Calibration
- https://github.com/kriswiner/MPU6050/blob/master/quaternionFilter.ino

- https://forum.arduino.cc/index.php?topic=445626.0
- https://forum.arduino.cc/index.php?topic=551063.0
- https://arxiv.org/ftp/arxiv/papers/1701/1701.07594.pdf
- https://learn.adafruit.com/ahrs-for-adafruits-9-dof-10-dof-breakout/magnetometer-calibration
- https://github.com/PaulStoffregen/NXPMotionSense/blob/master/OrientationVisualiser/OrientationVisualiser.pde
- https://learn.adafruit.com/ahrs-for-adafruits-9-dof-10-dof-breakout/using-adafruit-ahrs
- https://diydrones.com/profiles/blogs/advanced-hard-and-soft-iron-magnetometer-calibration-for-dummies?xg_source=msg_com_blogpost&id=705844%3ABlogPost%3A1676387&page=9
- https://www.lucidar.me/en/inertial-measurement-unit/mpu-9250-and-

arduino-9-axis-imu/

Librería para el datalogger shield

https://cdn-learn.adafruit.com/downloads/pdf/adafruit-data-logger-shield.pdf
https://learn.adafruit.com/adafruit-data-logger-shield

9. Motor principal

Su función es accionar el mecanismo de variación de la flotabilidad. El émbolo, con su posición dentro del cilindro cámara de flotabilidad, determina la flotabilidad. El motor propulsor, por tanto, tiene la función de cambiar la posición del émbolo. Para ello transformamos el movimiento rotativo en un movimiento lineal. Para esa función hemos decidido crear un tornillo de potencia a medida.

El motor hará rotar una tuerca, cuyo movimiento axial está impedido, y el tornillo asociado a esa tuerca se verá obligado a avanzar o retroceder axialmente.

En nuestros cálculos hemos establecido el tiempo requerido para una maniobra del émbolo, y la profundidad máxima de operación, que supone la máxima contra-presión que ha de vencer el motor actuador.

El producto fuerza axial x velocidad = potencia determina la potencia del motor.

Aplicando los rendimientos correspondientes (por rozamiento, tornillo, etc.) alcanzamos un requerimiento de potencia, y una velocidad de rotación.
Eso es lo que debe proporcionar el motor, aunque para ello puede que necesite una reductora adicional.

Hemos estimado una potencia de 30 W, con un tiempo de maniobra de 6 segundos, para avanzar 6 cm con una fuerza de 7.5 kg para trabajar a profundidad de H = 5 m.

Buscando motores DC disponibles en comercio electrónico ebay, etc, hemos encontrado dos alternativas de motor que pueden valernos.

- RH 158-24-250M,

- Nichibo 24Vdc HS5B 30:1

Uno es mayor que el otro, y sugiere mayor intensidad y mayor par.

El pequeño, por su parte tiene una mayor ratio de reductor.

Ambos tienen una tensión nominal de 24 V, y la correlación inversa de tamaño y ratio de reducción sugiere que pueden tener un rango de aplicabilidad solapada, aunque partiendo de distintas cualidades (sin reductora).

Para saber cómo deben interpretarse los datos de un motor eléctrico DC de este tipo, podemos consultar la referencia siguiente, que es muy completa: https://medium.com/luosrobotics/how-to-read-a-dc-motors-datasheet-f70fa440452b

A continuación, mostramos las fichas técnicas de los motores que tenemos:

DATI TECNICI
TECHNICAL DATA

serie / series **RH158**

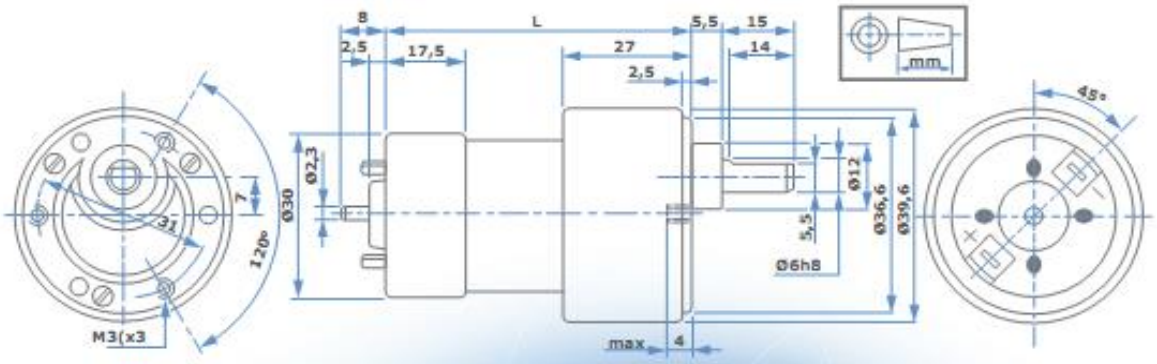
Soppressione disturbi con VDR sul collettore
Direzione di rotazione secondo polarità
Può essere montato in ogni posizione
Massimo carico radiale: 50N
Massimo carico assiale: 10N
Temperatura di esercizio: -20°C/60°C
Peso approssimativo: 190g

VDR interference suppression on the collector
Direction of rotation depending on polarity
Can be mounted in any position
Maximum radial shaft load: 50N
Maximum axial shaft load: 10N
Temperature range: -20°C/60°C
Approx weight: 190g

Valori tipici a temperatura ambiente +20° Tolleranza +/- 10%
Typical values at ambient temperature +20° Tolerance +/- 10%

TIPO / TYPE	TENSIONE NOMINALE / NOMINAL VOLTAGE	L	RAPPORTO :1 / RATIO TO:1	COPPIA NOMINALE / NOMINAL TORQUE	VELOCITÀ / SPEED		CURRENT / CORRENTE	
					SENZA CARICO / NO LOAD	CON COPPIA NOMINALE / AT NOMINAL TORQUE	SENZA CARICO / NO LOAD	CON COPPIA NOMINALE / AT NOMINAL TORQUE
	V	mm		Nom	rpm		mA	
RH158-12/24-15	12 24	84	14,14	10	440	300	<140 <70	660 330
RH158-12/24-30	12 24	84	29,75	20	210	140	<140 <70	660 330
RH158-12/24-75	12 24	86,5	78,84	50	91	55	<140 <70	680 340
RH158-12/24-100	12 24	86,5	94,37	60	66	45	<140 <70	680 340
RH158-12/24-200	12 24	89	199,5	100	33	23	<140 <70	580 290
RH158-12/24-250	12 24	89	243,9	100	28	21	<140 <70	500 250
RH158-12/24-510	12 24	72	512,95	100	12	10,5	<140 <70	300 150
RH158-12/24-830	12 24	72	829,92	100	10	9	<140 <70	270 135

RH158



RH158

micro motors s.r.l.
Viale Piave, 80/82 - 23879 VERDERIO (LC) ITALY
Tel. 039.510611-499 Fax 039.513617
www.micromotors.eu - info@micromotors.eu

Ilustración 8 Ficha técnica de uno de los motores estudiados 1. Micromotors.

En nuestro caso, dispone de una reductora planetaria incorporada con ratio 250.

Su tamaño moderado, y su alta relación de reducción indica que es un motor de muchas rpm, bajo par, baja intensidad. Su tensión nominal son 24 V. Previsiblemente lo alimentaremos a menor tensión par acomodar nuestra potencia e intensidad.

Entre nuestras limitaciones en la instalación eléctrica está la intensidad, por lo que este motor puede ser una opción mejor que otros de menor tensión que tendrían que dar la potencia requerida a mayor intensidad.

Valori tipici a temperatura ambiente +20° Tolleranza +/- 10% — Typical values at ambient temperature +20° Tolerance +/- 10%

TIPO TYPE	TENSIONE NOMINALE NOMINAL VOLTAGE	L	RAPPORTO :1 RATIO TO:1	COPPIA NOMINALE NOMINAL TORQUE	VELOCITÁ SPEED		CURRENT CORRENTE	
					SENZA CARICO NO LOAD	CON COPPIA NOMINALE AT NOMINAL TORQUE	SENZA CARICO NO LOAD	CON COPPIA NOMINALE AT NOMINAL TORQUE
	v	mm		Ncm	rpm		mA	
RH158·12/24·15	12 24	64	14,14	10	440	300	<140 <70	680 330
RH158·12/24·30	12 24	64	29,75	20	210	140	<140 <70	680 330
RH158·12/24·75	12 24	66,5	78,94	50	91	55	<140 <70	680 340
RH158·12/24·100	12 24	66,5	94,37	60	68	45	<140 <70	680 340
RH158·12/24·200	12 24	69	199,5	100	33	23	<140 <70	580 280
RH158·12/24·250	12 24	69	243,8	100	26	21	<140 <70	500 250

TIPO TYPE	TENSIONE NOMINALE NOMINAL VOLTAGE	L	RAPPORTO :1 RATIO TO:1	COPPIA NOMINALE NOMINAL TORQUE	VELOCITÁ SPEED		CURRENT CORRENTE	
					SENZA CARICO NO LOAD	CON COPPIA NOMINALE AT NOMINAL TORQUE	SENZA CARICO NO LOAD	CON COPPIA NOMINALE AT NOMINAL TORQUE
	v	mm		Ncm	rpm		mA	
RH158·12/24·250	12 24	69	243,8	100	26	21	<140 <70	500 250

Ilustración 9 Datos seleccionados de uno de los motores estudiados. Micromotors.

Pot = V.I = 24 x 250 V.mA = nominal = 24 x 0.250 = 6 W nominal.
Se asume que nominal indica a máxima eficiencia (rendimiento).
Par nominal x rpm nominal = 100 x 21 N.cm.rpm
rpm nominal/rpm sin carga = 21/26 == 21 * 0.04 == 0.84

Ese valor en % nos indica el punto de funcionamiento de máximo rendimiento,

Sin carga (_nl):
pot_nl = V.I_nl = 24 x 70 mA = 6/3.5 = 1.75 W consumido en rozamiento y perdidas internas.

Se estima que el rendimiento puede ser del orden de:
(6-1.75)/6 = 70%.
La potencia util será 6 x .7 = 4.52 W
Ese motor va casi perfecto para nuestro caso, en potencia, Pero en rpm es demasiado lento: 21 rpm.

En nuestro tornillo de potencia deseamos recorrer unos 60 mm en 6 segundos: 10 mm/s, que con un paso 0.5 mm supone dar 20 rev/s, o 20 x 60 = 1200 rpm.

Necesitaríamos cambiar la salida con una multiplicadora x 60, que pueden ser 2 etapas de x8 de un tren planetario adicional, que deberíamos construir.

Otra posibilidad es intercambiar la caja reductora con otro motor que tenemos, cuya reductora es de x250.

Veamos

Reductor / 250 + multiplicador x 60 == x 60 /250 = = 1 / 4 (reductor).

Si cambiamos la reductora /250 que viene con ese motor por otra reductora/ 30.
Bastará una multiplicadora externa tal que:

1 / 4 == 1/ 30 x M, de donde se despeja M == x 7, el elemento externo es una multiplicadora x 7, que está a nuestro alcance en forma de tren planetario, estudiado en otro apartado.

Al hacer girar a mano (en vacío y separadas de sus motores) ambas reductoras de x30 y de x250 se aprecia la diferencia de rozamiento que supone los escalones de reducción adicionales que las diferencian.

El formato externo de ambas reductoras es similar, y admiten el mismo acoplamiento. Por eso, cuando el motor es de tipos diferentes a ese formato, requieren un disco de acoplamiento. En nuestro caso, el motor 27D ya tiene es acoplamiento, con lo que cambiar una reductora por otra no tiene problema.
https://www.micromo.com/technical-library/dc-motor-tutorials/motor-calculations

La reductora que incorporan estos micro-motores es de tipo SPUR (trenes compuestos de ejes paralelos) (NO es de tipo planetario con varias etapas en serie).

En concreto este reductor es desmontable respecto al motor, y desmontado tiene el siguiente aspecto:

El aspecto del despiece de un tren planetario es de otro tipo, como se ve en la figura siguiente, sacada de Gearhead-Construction-and-Use-White-Paper-MICROMO-new-1.pdf

https://www.micromo.com/technical-library/frequently-asked-questions

En ese grafico de la pag5 del pdf, DC_Motor_Calculations.pdf. Se aprecia un ejemplo de las curvas propias de un motor como el nuestro.

La potencia en el punto de máximo rendimiento es del orden de:
0.85. rpm vacio * 0.15 * par stall =

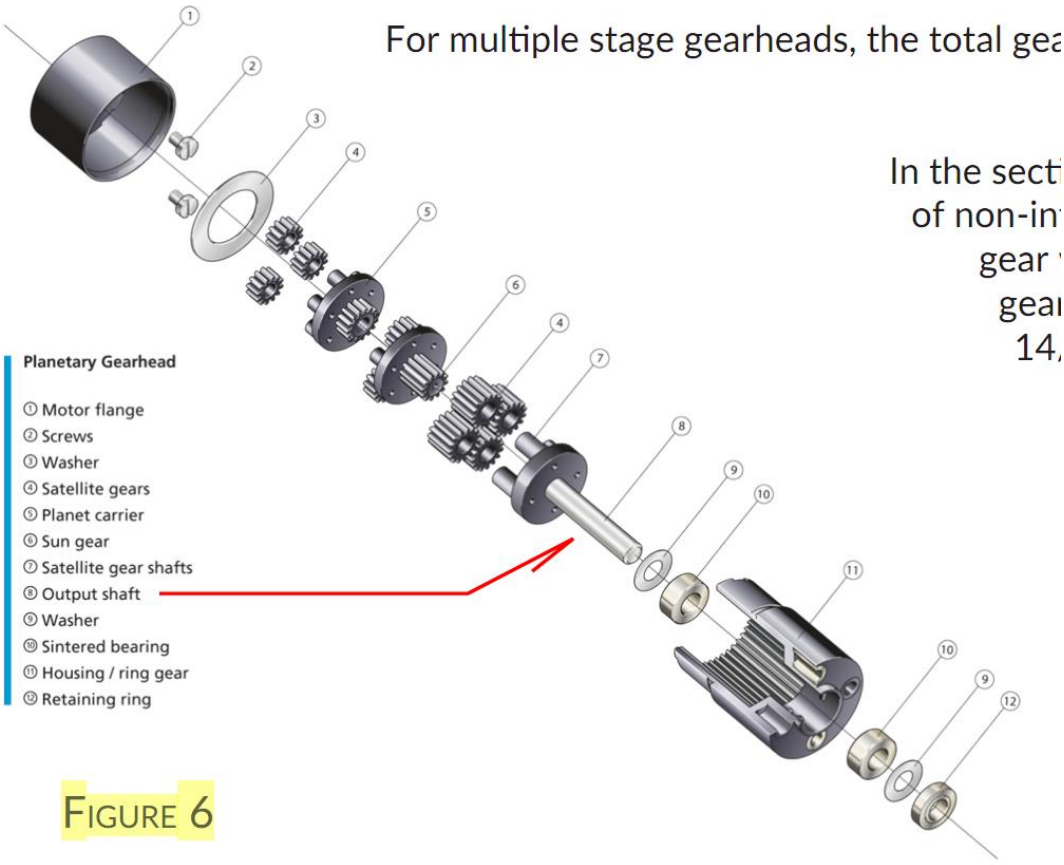


Ilustración 10 Despiece de un reductora de engranajes planetario. Micromo.

Examinemos ahora el otro motor, del cual no tenemos todos los datos, sino solo algunas indicaciones.

Ambos motores tienen una resistencia de rotor, medida entre ls escobillas, de 30ohm, en muchos casos, y en otros creciendo hasta 60 ohm. Esas variaciones según la posición angular tienen que ver con el número de polos y las delgas.

Debe haber una relación entre V I R en rotor bloqueado: V = r.I, de modo que I = V/30 A, así para v = 15 V, resulta I =0.5 A, mucha intensidad, pude dañar las delgas, escobillas, etc.

En su lugar se alimenta a tensión reducida, y se mide la resistencia, para comprobar que cumple la previsión.

para v = 1 V, I = 0.033 A

Ambos tienen 6 polos (3 pares). Y están marcados como CCW. De los 6 hilos o pines, el grupo de 2 es para la alimentación del motor y el otro grupo de 4 para el encoder (que a su vez son 2 para alimentación, y los otros 2 de salida de cada uno de los 2 sensores A, B),

1. El motor grande se caracteriza por 37D mm, diametro, y pesa 260 gr completo (con encoder y reductora x30). He encontrado en pdf en alibaba, alguno parecido puede ser el de pololu es de 12 v y me permite calcular la R interna, y el de alibabab de 24v, aunque no detalla datos.

2. El motor pequeño (menor) es 27D mm, y pesa184 gr

Entre las opciones disponibles, se elige el motor grande, pero desmontando la reductora. En esas condiciones tendremos unas rpm y un par que creemos que pueden accionar directamente la carga a buena velocidad, sin sobrecarga.

9.1 Ensayos de motor principal.

Finalmente, tenemos disponible dos motores parecidos para que actúen como motor principal.

- RS.555SH, DD433821, sin reductor, y sin encoder.
- Otro, con encoder magnético.

Solamente el de encoder magnético nos resultará útil, puesto que podemos valorar su velocidad y posición.

Sin embargo, de ninguno de los dos motores conocemos su ficha de características, por lo que deberemos hacer algunos ensayos para conocerlos. Como intuimos que son similares, los ensayos de ambos serán mutuamente contrastables y validantes.

Usaremos una fuente de alimentación estándar de PC, capaz de dar potencias de 100 W y 300 W a unos 5 V y 12 V respectivamente.

En concreto, en vacío las tensiones de la fuente son (medidas con polímetro):

- 5.28 v,
- 11.46V.

Como la potencia de los motores es muy inferior a la capacidad de la fuente, se constata que la tensión en línea no cae cuando se conectan cualquiera de los motores.

En esas condiciones se han hecho mediciones en vacío (sin par resistente al eje del motor), y se obtienen rpm al eje, e Intensidad absorbida, medida con polímetro intercalado en el circuito de CC.

Lo mismo para 12 V nominales, adicionalmente, pero solo a la mínima tensión, 5V se han medido las intensidades a rotor bloqueado, (durante apenas 1

segundo, para evitar calentamiento de los bobinados).

Para la medición de las rpm en vacío se ha usado una aplicación para teléfono móvil (tipo smartPhone) que simula una lámpara estroboscópica, con capacidad de captar la frecuencia del sonido emitido por el eje en rotación, y replicar esa señal la como luz pulsada a la misma frecuencia, que además se indica en la pantalla medida en Hz. Con el alumbrado estroboscópico del eje en rotación se puede comprobar visualmente que la medición es realista.

La aplicación se llama Strobily Free v2.0.0 By Bryan Coventry aka thepenguin77, y ha sido muy eficaz en su cometido.

En el siguiente cuadro se resume los datos obtenidos en los ensayos.

		nichibo 24Vdc HS5B 30:1, (sin reductora, pero con encoder)		RS.555SH, DD433821, sin reductor, y sin encoder.	
Vref		5	12	5	12
Vreal		5.28	11.46	5.28	11.46
Giro libre, en vacío	rpm	1175	2926	687	1710
	I	0,105	0.115	0.060	0.065
	Resist aparente (V/I)	50	99	88	176
	Rpm/V	222	255	130	149
Rotor bloqueado	I	0.350	N/a -	0.300	N/a -

La principal diferencia entre ambos motores es que uno de ellos tiene aproximadamente el doble de potencia que el otro, y eso se materializa en

rpm, e intensidad. En uno de ellos es del orden que el doble del otro. El de más intensidad da más velocidad (rpm).

El del encoder tiene en su propio nombre una indicación de que soporta 24 V, mientras que del otro motor no tenemos ese dato. La forma de conocer cuál es el límite de potencia o tensión en él sería por medio de un ensayo de resistencia. Ir aumentando la tensión en escalones pequeño, comprobar la temperatura estacionaria que se alcanza en cada uno de esos escalones de carga, y limitar la temperatura perceptible (la de la carcasa) a valores correlacionados con los de las bobinas, y admisibles. Típicamente en las bobinas no se debería superar los 50º, y en la carcasa no se debería pasar de 35º.

Lo más relevante de este cuadro, en este momento, es que las intensidades están acotadas a valores que podremos gestionar con hardware de tipo DIY, de precio moderado y accesible, como el del entorno Arduino. Eso es lo que pretendemos, pues suele existir mucha documentación y explicaciones alrededor de esas plataformas.

El siguiente paso es valorar si podemos gestionar ese motor elegido con el shield del que disponemos:

- Motor control shield for arduino, DK electronics.
- Sus características están descritas en el siguiente pdf.
- Shield de motores con L293D (L293D Motor shield)

0.600 A máximo por cada canal de motor, hasta 4 motores. Eso nos permite mover pares ligeros. Si se requiriese más habría que añadir más puentes h en paralelo, soldados modo piggy., o cambiar los integrados por otros de mayor potencia. Eso sería posible, pues van montados en zócalos. Lo otro también.

Volviendo al guión principal del proyecto, los siguientes pasos van a ser:

- Preparar un ejemplo de gestión de motores DC desde arduino, a través de la placa motor Shield (DK electronics v10, que ya tengo). Usar algún tutorial donde se indiquen las librerías, el code, etc.

- En nuestros ensayos usaremos una alimentación externa (distinta de la de placa Arduino) para los motores, y que será a 5 V para minimizar consumos y riesgos. El motor que usaremos será el 550 que no tienen encoder.

- Hemos leído sobre problemas con alimentadores BOOSTers elevadores de tensión DC-DC, al mover motores DC. Tendremos que hacer una prueba por separado, tester tensiones,… y después alimentar al motor.

    - Vin 5.27, Vout 12.23, en bornas del booster sin carga.

    - Alimentando al motor 550, en vacio: Iin motor 0.065A, Vin motor 11.71, V out booster 12.47; Vin booster5.02 Vout fuente 5.21, rpm 1768. Se nota que la perdida en los cables de alimentación es sustancial, en apenas 20-40 cm en cada tramo. La potencia perdida en esos tramos es: P = V.I = 0.7*0.065 = 0.045 W = 45 mW a la salida del booster, y similar (suponemos) a la entrada. Tiempo despues (45'), midiendo finamente con el estroboscopio (hasta estabilizar la imagen nítida), resultan rpm 1728, en régimen estacionario de temperatura T 30.

    - Medimos intensidad a la entrada al booster I 0.17 A a la misma escala que las medidas anteriores. (Imax 15A). Luego la perdida de potencia en este tramo será P = VI = 0.19*0.17 = 32 mW.

    - También podemos estimar el rendimiento de esa conversión del booster: Pin = VI = 5.02*.17= 0.852W, Pout= VI = 12.47*0.065= 0.810; rto= Pout/Pin=0.810/0.852 = 0.95 = 95%. El rto del booster a esa carga es muy bueno, 95%.

- En esas condiciones, al cabo de un tiempo, 15`, la temperatura máxima en la carcasa se detecta en un punto, marcado T, a mitad de la longitud del cuerpo del cilindro, a 90º con la etiqueta de papel. Con el sensor de temperatura de infrarrojos, tipo pistola, se miden 29.0 ºC. En este motor, las toberas de ventilación de la carcasa están cerradas con una chapa móvil (removible). Otros 5' después, se mide 30.3ºC, y como punto ambiente de similar reflectancia 25.6ºC

- Otros 10' despues volvemos a medir T 30.1 en el punto de referencia. Las mínimas diferencias que se han encontrado en ese tiempo se deben a un muy tenue movimiento de aire ambiente, y que ls temperatura en la carcasa no es uniforme. Hay algún punto más caliente, y el gradiente es fuerte. Cuesta localizar la temperatura máxima 'real', pues está localizada muy puntualmente.

- Se asume que la temperatura estacionaria de la carcasa a ese régimen es T 30 en el punto caliente indicado, cuando la ambiente de referencia es T 25.6. Eso se ha contrastado varias veces en periodos más amplios 60'….

- La conclusión de esta fase es que la alimentación con booster del Dcmotor es muy eficaz, y permite amplio rango de tensiones aplicables al motor.

- Descripción del BOOSTER

    - Es una placa denominada XL6009, DC-DC converter step up, 4A 60V400khz
    - Admite hasta 4A de corriente, y el rto puede llegar a 95%. Es un elevador de tensión, de modo que la salida ha de ser mayor que la entrada. La entrada puede ser de 3 a

30V.

- La placa que tengo era del 2015, aunque el datasheet lo he localizado ahora, y puede que no sea el del modelo original, aunque no puedo saber más de momento.

- Ese motor nos va a permitir hacer algunos ensayos sobre la propia estructura motriz del submarino, a una velocidad con carga, derivada de partir de vacío (sin carga) a unos 700 rpm, y con un consumo de unos 0.060 A

- Hay que habilitar la funcionalidad de los fines de carrera de protección al recorrido del émbolo de la cámara de flotabilidad.

Deberá ser una protección de hardware y de software, pero que no inhabilite la autonomía del AUV.

9.2 Prueba del motor desde el motor Shield con Arduino, con código

1- Sigo con el guion, busco una librería para el motor shield v1:
https://learn.adafruit.com/adafruit-motor-shield/library-install
https://github.com/adafruit/Adafruit-Motor-Shield-library/zipball/master

Arranco mi antiguo IDE (Arduino – Genuino 1.8.5) en asus negro, enchufo el USB del Arduino, y tras reconocerse (automáticamente) el driver adecuado queda conectado en el COM8. La placa es reconocida como la MEGA ADK, porque yo se lo he marcado así. Pruebo a cargar un sketch demo ('fade') cambiando el pin, parámetro de software a 13, para que luzca el led de la placa (MEGA). Carga sin problema (compila en un tiempo razonable de unos 22 '', y otros 3'' en cargar) y se ejecuta ok, También modifico el delay y vuelvo a cargar, y vuelve a funcionar modificado. Parece que tengo la configuración correcta en el IDE y la placa. En pruebas sucesivas con leves cambios del delay se produce la carga y ejecución en unos 14''.

En esa prueba la placa ADK tenía añadidos el shield de DATA LOGGING, (con su bat para RTC), y encima el MOTOR shield. Ambos shields tienen sus pilotos verdes de ON operativos. También la MEGA ADK

2- ... y seguimos el guión de la web… sobre la alimentación de la placa, el jumper, la fuente externa específica para los motores, su polaridad, el led ON, ….
https://learn.adafruit.com/adafruit-motor-shield/power-requirements

3- Preparo la carga de la librería adecuada para el MOTOR shield. Para ello uso una buena funcionalidad de la v 1.8.5 del IDE de Arduino. El Gestor de Librerías. A partir de un browser ?on line' puede recorrer un repositorio de librerías disponibles, e instalar automáticamente cualquiera de las que se seleccione. Así cargo la de mi interés, l v10

de Adafruit motor shield, (ahora ya obsoleta por existir la nueva v2 tanto en placa como en librería). Nosotros necesitamos la antigua, coherente con esa placa v10.

4- En fecha 17-7-2018 he hecho pruebas con el sketch motor party, y he testado un servo y el DCmotor con buen resultado, en ambos casos. La velocidad del DCmotor está limitada por la tensión de la alimentación externa, y el programa puede modularla por debajo de esa tensión externa por medio de PWM, Pese a todo, algunos motores no giran con tensiones inferiores a 1.5V, como es el de mi caso SH550.

5- Lo siguiente es el ejemplo de código usado en el Arduino para probar el funcionamiento del motor, el servo, la placa motor shield, y el propio Arduino.

6- El código fuentes está escrito en un leguaje tipo C, propio del Arduino, y del entorno de PC que lo complementa.

7- Comentaremos que el código del ejemplo consta de:

- Comentarios,
- Inclusión de las librerías propias de gestión de Motor Shield, y de los servos.
- Declaración de elementos tipo:
    a. DCmotor, en particular llamado 2, y asignado a ese terminal M2
    b. Servo1, (aunque en realidad funciona en el terminal servo2 del motor shield
    c. Stepper, asignado de los terminales 2, y declarado de 48 pasos por vuelta. En mi caso no he usado ningún stepper, pero eso no anula el resto de la prueba.
- Después un bloque, que hace el setup o inicialización de servo y motor.

a. Se inicia un canal serie de comunicación a 9600 baudios, y se lanza por él el mensaje “fiesta de motores”.
b. El servo es asignado al pin de control 9
c. El motor es declarado para una velocidad 200, que debe leerse como200/255, en % de la velocidad nominal que se obtendría a la tensión de alimentación 100% de la fuente usada por el Motor Shield.
d. Fin del bloque de setup

- Después se definen dos parámetros:
  a. i, contador para bucles
  b. dly, tiempo de espera en ms antes de continuar la ejecución siguiente a la sentencia delay (dly).
- Y se inicia un bucle continuo donde se incluyen varias pruebas:
  a. Hacia adelante:
     i. En sucesivos pasos desde i =1 a 255
        1. Se hace avanzar el servo al paso i
        2. Se pone al motor a velocidad i
        3. Se mueve el stepper un paso
        4. Se espera (con todo eso funcionando) una fracción de tiempo dly ms, y se vuelve al siguiente valor i+1
     ii. Se repite otro bucle igual, con todos los movimiento hacia adelantes, pero reduciendo la velocidad o la posición, en pasos regresivos, desde i=255 hasta 1
  b. Ahora movimientos hacia a atrás, repitiendo las series
     i. En pasos sucesivos crecientes de i
        1. Servo
        2. Dc motor
        3. Stepper
     ii. En pasos sucesivos decrecientes de i
        1. Servo
        2. Dc motor
        3. Stepper
  c. Finaliza el bucle y se vuelve a repetir el loop indefinidamente.

8- Una vez descrito el código estamos más familiarizados con las sentencias y la sintaxis.

Añadimos algunas referencias adicionales para explicar el código:

```
// Adafruit Motor shield library
// copyright Adafruit Industries LLC, 2009
// this code is public domain, enjoy!

#include <AFMotor.h>
#include <Servo.h>

// DC motor on M2
AF_DCMotor motor(2);
// DC hobby servo
Servo servo1;
// Stepper motor on M3+M4 48 steps per revolution
AF_Stepper stepper(48, 2);

void setup() {
 Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Motor party!");

  // turn on servo
  servo1.attach(9);

  // turn on motor #2
  motor.setSpeed(200);
  motor.run(RELEASE);
}

int i;
int dly = 200;
```

```
// Test the DC motor, stepper and servo ALL AT ONCE!
void loop() {
  motor.run(FORWARD);
  for (i=0; i<255; i++) {
    servo1.write(i);
    motor.setSpeed(i);
    stepper.step(1, FORWARD, INTERLEAVE);
    delay(dly);
  }

  for (i=255; i!=0; i--) {
    servo1.write(i-255);
    motor.setSpeed(i);
    stepper.step(1, BACKWARD, INTERLEAVE);
    delay(dly);
  }

  motor.run(BACKWARD);
  for (i=0; i<255; i++) {
    servo1.write(i);
    motor.setSpeed(i);
    delay(dly);
    stepper.step(1, FORWARD, DOUBLE);
  }

  for (i=255; i!=0; i--) {
    servo1.write(i-255);
    motor.setSpeed(i);
    stepper.step(1, BACKWARD, DOUBLE);
    delay(dly);
  }
}
```

FIN del CODIGO ejemplo.

A partir de lo aprendido en ese código vamos a crear otro, adaptado para nuestro propósito:

Queremos que arranque desde parado, rpm =0, alcance la velocidad nominal RPMn en un periodo Tr, y se mantenga a esa velocidad durante un tiempo Tm.

Al terminar ese tiempo, hará el proceso inverso, descendiendo hasta velocidad 0, tardando para esa operación un tiempo Tr,

El ascenso en rampa de duración Tr lo dividiremos en 10 escalones.

El nuevo código quedará así:

```
// NAA, navegación arriba y abajo del AUV
// jun 2018
// this code is public domain, enjoy!

#include <AFMotor.h> // librería para el motorShield de Adfafruit v10
#include <Servo.h> // no necesaria salvo si se usan servos, mas adelante

// DC motor on M2
AF_DCMotor motor(2); // se habilita la conexión M2 de la motorShield
// DC hobby servo
Servo servo1;
// Stepper motor on M3+M4 48 steps per revolution
AF_Stepper stepper(48, 2);

void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  Serial.println("Posibles mensajes de gestion en el monitor de puerto serie...");
```

```
  // turn on servo
  servo1.attach(9);

  // turn on motor #2
  motor.setSpeed(200);
// SET speed, pese a su nombre, solo controla la tensión de alimentación, no las rpm.
// en este caso modulando (200) vía PWM la tension de la fuente (VN), lo alimentamos a 200/255 *VN .
// Las rpm que resulten dependen, además, del par resistente,…..
  motor.run(RELEASE);
}
int Tr = 5;    //tiempo de rampa de aceleración, segundos
int escalones = 255;    // escalones: 255, mejor si solo son 10
int Tm = 10;  // tiempo de velocidad mantenida, después de alcanzar el fin de la rampa.
int i;
int dly = Tr/escalones*1000; // dly = Tr/10*1000, en ms, 500,    asumimos escalones:10

// Test the DC motor, stepper and servo ALL AT ONCE!
void loop() {
  motor.run(FORWARD); // establece la polaridad de giro del DC motor
  // iniciamos la rampa de aceleración de 0 a VN
  for (i=0; i<255; i++) {         // se establecen 255 escalones sucesivos
    motor.setSpeed(i);  // en cada escalon se alimenta a una tensión i/255 VN, y resultarán velocidades rpmi, según el par resistente.
    servo1.write(i);
    stepper.step(1, FORWARD, INTERLEAVE);
    delay(dly);   // para asignar que en cada escalón se tarde un tiempo de operación controlado (por nosotros), se aplica un lapso dly ms
  } // aquí concluye el bucle de rampa creciente.
// empiezo otro bucle que dure Tm, a velocidad constante, la maxima
  for (i=0; i<Tm; i++) {          // se establecen Tm escalones sucesivos 1000 ms
```

```
   motor.setSpeed(255);   // en cada escalon se alimenta a una tensión
255/255 VN, y resultará velocidad rpm, según el par resistente.
   servo1.write(i);
   stepper.step(1, FORWARD, INTERLEAVE);
   delay(1000);  // cada vuelta de este bucle tarda 1 segundo aprox
 } // aquí concluye el tiempo de tensión  VN constante durante Tm segundos.

        // ahora aplicamos una rampa de deceleración hasta parada
for (i=255; i!=0; i--) {
   motor.setSpeed(i);
    servo1.write(i-255);
    stepper.step(1, BACKWARD, INTERLEAVE);
   delay(dly);
        } // aquí concluye el bucle de rampa decreciente hasta tensión V = 0.
     }   // y aquí cierra el loop principal void,
```

Todo lo anterior durará Tm + 2*Tr, y es un proceso con arranque y parada sAUVe, sin golpes de aceleración, en la que se alimenta al motor en un sentido de avance, alcanzando la máxima tensión por el camino.

Eso puede mover el émbolo de inmersión desde la posición central (0) a la posición de avance que se alcance en ese tiempo, (o se limite por los fines de carrera).

Otro proceso parecido, pero con tensión invertida (motor.run(BACKWARD);) servirá para hacer retroceder el émbolo desde una posición avanzada.

10. Motor del husillo de contrapeso para trimado.

Para esta función podemos usar un DC motor pequeño, con un control similar al del motor principal.

Alternativamente, también podemos usar un motor de tipo stepper, dependiendo de la disponibilidad.

Ni el par, ni la velocidad son especialmente críticas, ya que para su función las exigencias son bajas.

11. Servo para gobierno

Para la función motorizada de gobierno usaremos un servo tipo S3003, por su disponibilidad comercial, economía, y suficiente capacidad para mover el contrapeso escorante.

Con esto queda completada la descripción de los órganos motores del AUV: El motor (principal) del control de propulsión, y el motor de maniobra (genera escora, puede ser un servo), así como el husillo motorizado adicional para corregir el trimado fino inicial en cada misión.

12. Sensores

12.1 Detector Fin de Carrera.

Un final de carrera es un pulsador (interruptor eléctrico con muelle antagonista) que se suele utilizar en sistemas automatizados para delimitar la acción de cierto elemento con movimiento físico, en este caso un motor.

Tienen un par de características, que los diferencian de un pulsador normal: la primera es que suele disponer de una pequeña palanca o adaptador que transmite el movimiento del objeto en seguimiento hasta el pulsador eléctrico. La otra característica, es que constan de 3 pines (los más comunes), identificados como COM (común), NO (normalmente abierto), NC (normalmente Cerrado), esto ¿qué significa?:

NO, se refiere a que cuando el pulsador no está accionado la corriente no pasa, y cuando se pulsa pasa. (NormalyOpen, Normalmente Abierto, NO,NA) NC, justo al contrario. Cuando el pulsador no está accionado la corriente pasa, y si se acciona deja de pasar (NormalyClosed, Normalmente Cerrado).

COMÚN, es el pin común en cualquiera de las dos posibles configuraciones. El común lo usaremos siempre y, dependiendo de la configuración que queramos, usaremos NC o NO.

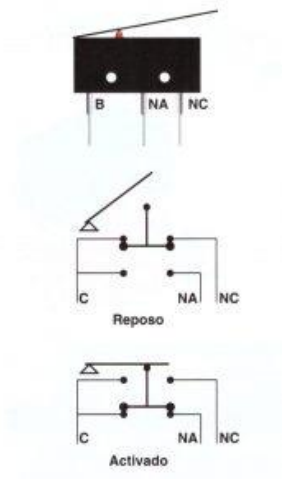
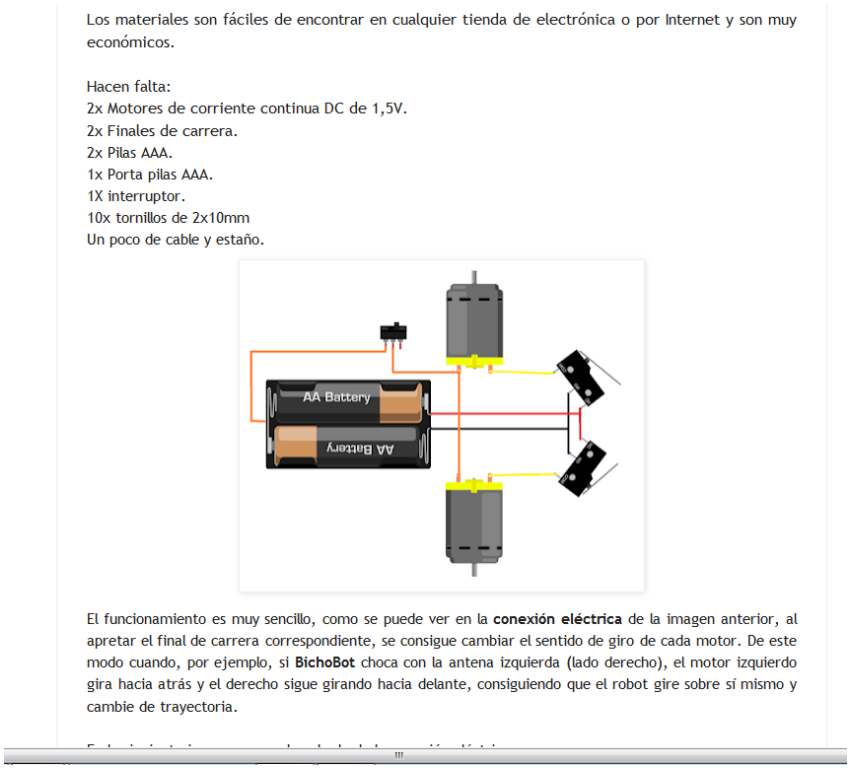


Ilustración 11 Fin de carrera. Sr Ferrete.

12.2 Gestión de los detectores FIN DE CARRERA

También se debe implementar una gestión inteligente de los fines de carrera de embolo, etc., para garantizar la seguridad de los motores. Esto es necesario en el motorDC y en el stepper, En el caso del servo, el propio dispositivo incorpora sus protecciones


Los materiales son fáciles de encontrar en cualquier tienda de electrónica o por Internet y son muy económicos.

Hacen falta:
2x Motores de corriente continua DC de 1,5V.
2x Finales de carrera.
2x Pilas AAA.
1x Porta pilas AAA.
1X interruptor.
10x tornillos de 2x10mm
Un poco de cable y estaño.

El funcionamiento es muy sencillo, como se puede ver en la **conexión eléctrica** de la imagen anterior, al apretar el final de carrera correspondiente, se consigue cambiar el sentido de giro de cada motor. De este modo cuando, por ejemplo, si **BichoBot** choca con la antena izquierda (lado derecho), el motor izquierdo gira hacia atrás y el derecho sigue girando hacia delante, consiguiendo que el robot gire sobre sí mismo y cambie de trayectoria.

Ilustración 12 Motores con limitador de carrera. Sr Ferrete.

http://puspis.blogspot.com/2013/01/control-de-motor-cc-mediante-2-senales.html

En la práctica podemos añadir un par de interruptores limitadores de fin de carrera (limit switch) para evitar que el motor gire más allá de unos límites físicos prefijados. De tal manera que cuando se alcanza uno de los límites, se abre el circuito gracias al interruptor y sólo será posible el giro en sentido opuesto. Igualmente ocurrirá al alcanzar el límite del sentido contrario. Esto se consigue gracias a unos simples diodos que sólo conducen la corriente en

el sentido adecuado.

En la siguiente imagen vemos los seis casos posibles.

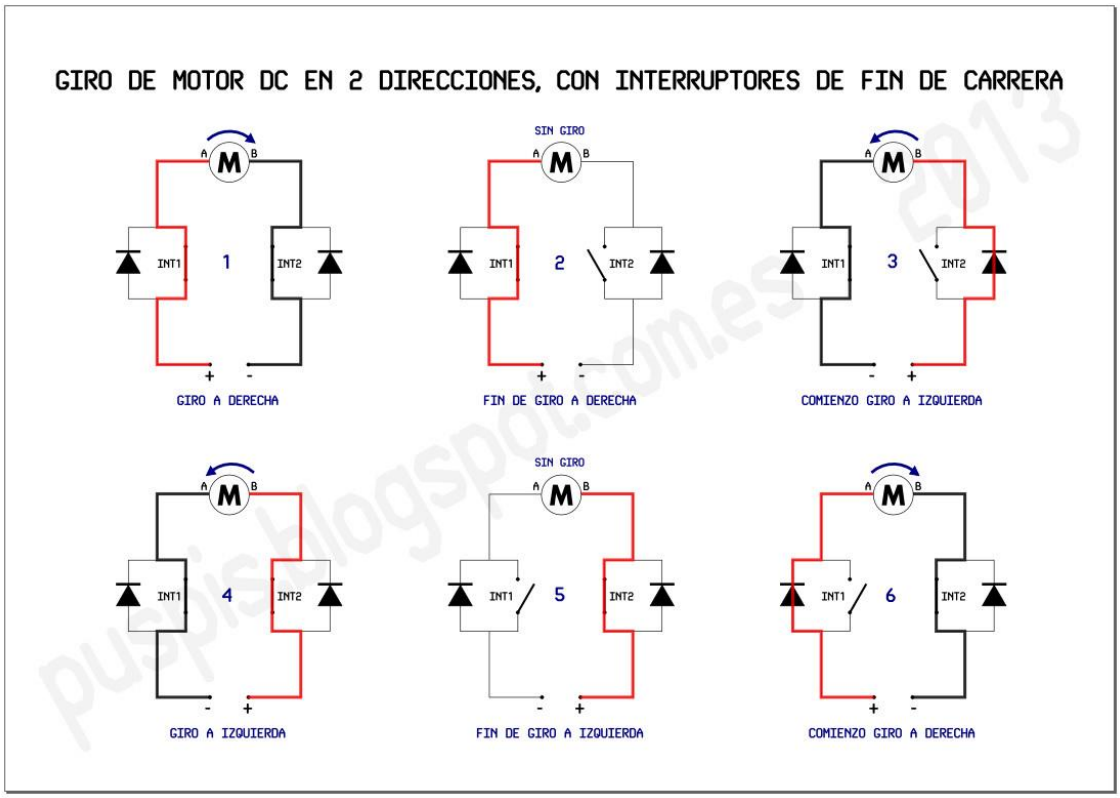


Ilustración 13 Giro motor DC en 2 Direcciones. Sr Ferrete.

Los sensores fin de carrera para Arduino, con placa, incorporan una circuitería que aporta algunas cualidades de interés al montar conexiones

http://srferrete.es/finales-de-carrera-para-tu-cnc-basado-en-grbl

Por lo general en máquinas de responsabilidad, la configuración por la que se suele optar es por Normalmente Cerrado, ya es la más segura en caso de que nuestro cableado se rompa. Si tenemos nuestra CNC con finales en NO y hay una cable roto cuando el carro pulse dicho botón, Arduino no se va a enterar y la máquina no se parará. Se suelen poner para cada motor 2 fc, uno en cada extremo de su recorrido.

En el dibujo que sigue los 2 finales de carrera están en serie, de modo que

actuar sobre cualquiera de ellos da el mimo resultado, son indistintos. Su significado debe interpretarse del contexto en que se produce:

- Si el motor avanza en X+, y se toca el FC+, todo es coherente, y el Arduino indicará que se termine el movimiento hacia X+.
- Si el motor avanza en X+, pero por el contrario se toca el FC- (el del otro extremo, hacia X-), el Arduino debe parar el movimiento en curso hacia X+. Puede interpretarse que no era previsible que saltara es FC, y puede indicar QUE HAY UN MALFUNCIONAMIENTO DEL FC
- Si los FC de los dos extremos están en serie, nunca se interpretará mal funcionamiento por este último motivo

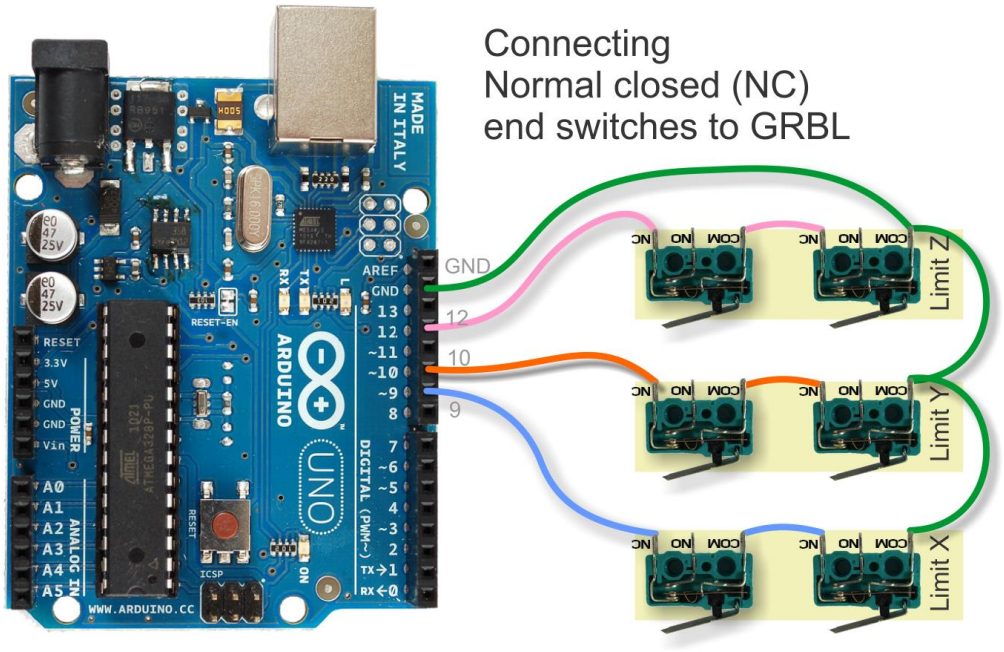


Ilustración 14 Conexiones fines de carreras conectadas en un Arduino. Sr Ferrete.

Esos finales de carrera no cortan la alimentación del motor, sino que dan una señal al Arduino para que haga lo que 'corresponda'. En cada caso la decisión depende de lo que estuviera ocurriendo.

Usualmente puede ser:

- Parar el avance en curso X+
- Iniciar avance opuesto X-. (con duración limitada)
- Parar después de un lapso de 1000 ms, o hasta que se vuelva a liberar

el FC

Eso sirve para desenclavar el eje, desactivar la señal de fin de carrera alcanzado, y dejarlo listo para el próximo movimiento (en cualquiera de las dos direcciones).

En cualquier caso, el disparo de un fin de carrera es un caso típico en el que debe usarse una interrupción del procesador, como también ocurría con el encoder.

12.3 Transductor de presión.

En fecha 15-7-2018 he encargado en aliexpress 2 transductores de presión para líquidos, que cuestan 9 eu/cu, y son de 5V, y miden hasta 1.2 Mpa, (unos 12 bar, unos 120 mca (m de altura de columna de agua).

https://es.aliexpress.com/item/DC-5V-Pressure-Transducer-Sensor-G1-4-0-1-2MPa-For-Oil-Fuel-Diesel-Gas-Water/32812346950.html

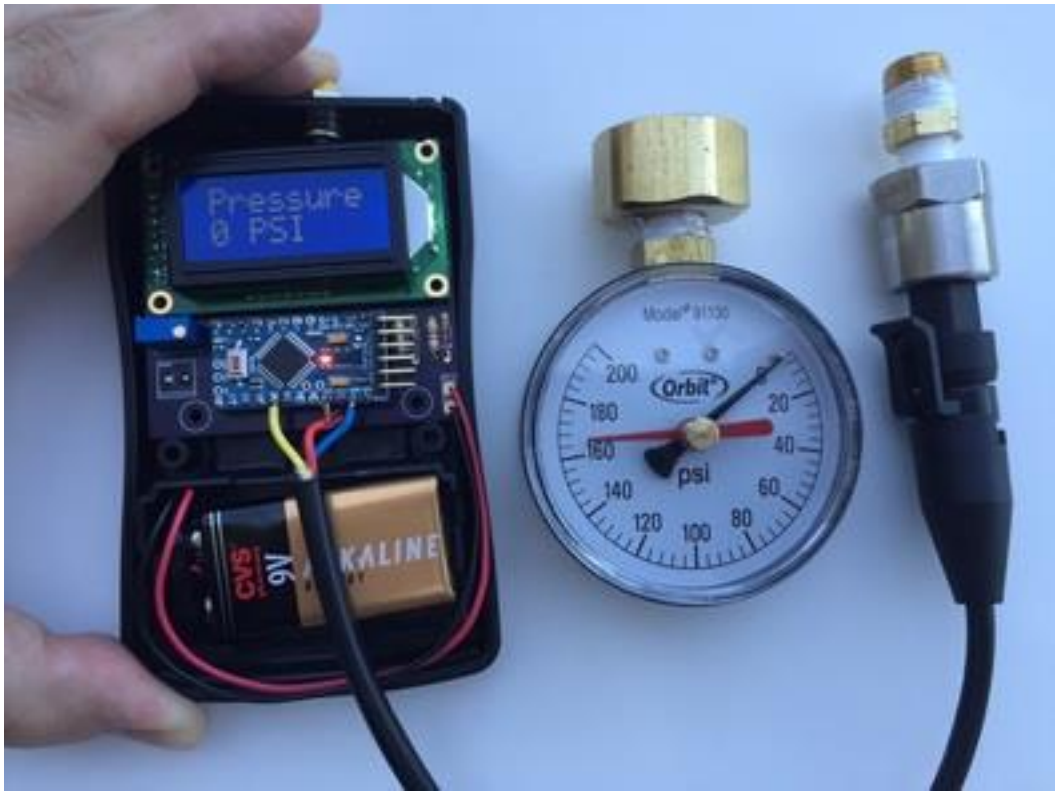


Ilustración 15 Sensor presión hidrostático. Ali Exprés.

Con ellos puedo crear un par diferencial de lecturas de presión (absoluta) que me permitirá estimar la velocidad de navegación via pitot, y a la vez la profundidad. Para completar, puedo medir el ángulo de trimado con otro sensor, el giróscopo, que incluye brújula magnética…, para la orientación NS.

En fecha 24-7-2018 (solo 9 días después) he recibido los sensores en la oficina de correos. Adicionalmente he recibido varios mensajes en el móvil sobre la evolución de la entrega del pedido a domicilio. Los sensores son

moderadamente grandes. Pero podemos usar un acoplador de rosca 1/2'' a otras menores, para adaptar a unas tuberías metálicas finas (diam 4 mm) que atraviesen el casco por la tapa, y vayan hasta el punto del exterior donde se desea tomar la medida de presión (remanso, Venturi, etc.).

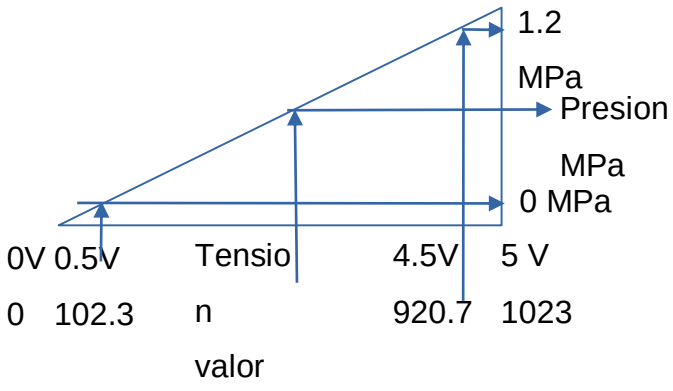
El siguiente enlace aporta una demostración de uso del sensor SKU237545, que incorpora este transductor. Incluye un ejemplo de código para chequearlo con Arduino, que podremos readaptar a nuestra necesidad.

Connection and Programming of SKU237545 Pressure Sensor.
https://www.youtube.com/watch?v=AB7zgnfkEi4
El código de origen tiene alguna errata, y además el sensor estaba mal calibrado_

- a presión atmosférica nivel del mar, V=v0
- mínimo 0.5V - 0 bar
- 4.5 V - 1.200.000 Pa

La calibración correcta sería según el siguiente diagrama, por semejanza de triángulos, asumiendo la linealidad de las lecturas de tensión del sensor:

Tensión (0-5V) → valor (0-1023) →
valor → presion



Presion/1.2 =(tension-0.5)/ (4.5 -0.5)= (valor-102.3)/(920.7-102.3)

(1 atm = 1013 mbar = 1.013 bar

lectura V	fondo escala 0-1023	Presion rango sensor 0-1.2MPa
0	0	-
0.5	1023*0.5/5 = 102.3	0
4.5	1023*4.5/5 = 920.7	1.2MPa
5	1023	-
V	val_sensor= 1023*V/5	pres = 1023*(V-0.5)/5

Un código que se ha preparado, probado y funciona, dando la profundidad en m bajo la superficie del mar (asumiendo agua de densidad 1.0 t/m3) sería el siguiente:

```
//presuresensor.ino
void setup() {
  Serial.begin(57600);
}
void loop() {
  int sensorVal = analogRead(A1);
//el valor leido en el pin de entrada está en la escal 0-1023.
// el valor 1023 sería cuando la tensión en aquel pin es el valor V con el que se alimenta la placa arduino.
// normalmente 5 V
// se constata que el sensor de presión mide respecto a la presión atmosférica estandar,
// para la cual la lectura de presion es 0. (p-p_standar)
// float voltage = (sensorVal * 5.0) / 1024.0;
  float voltage = ((sensorVal -102.3)/(920.7-102.3)) *4 +0.5;
   //float pressure_pascal = (3.0 * ((float)voltage - 0.47)) * 1000000.0;
  float pressure_Mpascal = (1.2 * ((float)voltage - 0.5)/4) ;
  float pressure_bar = pressure_Mpascal ;

  Serial.print("Sensor Value: ");
```

```
  Serial.print(sensorVal);

  Serial.print(" Volts: ");
  Serial.print(voltage);

  Serial.print("Pressure_MPa = ");
  Serial.print(pressure_Mpascal);

  Serial.print(pressure_bar);
  Serial.print(" bars");
  Serial.print("Profundidad bajo el nivel del mar");

  Serial.print(pressure_Mpascal*10);
  Serial.println(" mca");

//  Serial.print("Pressure = ");
  delay(200);
}
```

En la web se pueden encontrar otros ejemplos que corroboran que el enfoque que estamos haciendo es correcto:

https://www.dfrobot.com/wiki/index.php/Gravity:_Water_Pressure_Sensor_SKU:_SEN0257

Además, después de las pruebas y la calidad y poco ruido de las lecturas, se llega a la conclusión de que el sensor dá una buena resolución, de menos de 1 cm de columna de agua en las lecturas de profundidad. Eso también nos va a permitir prescindir de la segunda unidad para medir la velocidad, ya que podemos usar solo una medición de la velocidad vertical del AUV, y corregir la estimación de avance con otros parámetros, como el ángulo de trimado, etc.

12.4 Sensor inercial – IMU

Una unidad de medición inercial o IMU (del inglés inertial measurement unit), es un dispositivo electrónico que mide e informa acerca de la velocidad, orientación y fuerzas gravitacionales de un aparato, usando una combinación de acelerómetros y giróscopos. (Usaremos un acelerómetro de 3 ejes 3D, con sus respectivos giróscopos a los mismos 3 ejes, y un magnetómetro también a 3 ejes. Todo ello constituye el corazón de sensores de un navegador inercial. La navegación inercial, o por estima, permite orientarse de manera autónoma en ausencia de referencias posicionales externas.

Nuestro sensor está configurado en un pequeño circuito impreso que lleva varios integrados, y se comercializa con el nombre GY 9150, y tiene el chip MPU-9150. Nosotros usaremos un MPU9250, que es una versión posterior, mas perfeccionada. Aunque son compatibles en la mayoría de casos, el 9250 tiene mejores prestaciones por incluir un procesador de fusión de datos en el propio chip, lo que mejora la coherencia entre los diversos sensores. Mantiene un precio muy atractivo, unos 3euros en ebay/aliexpress.

https://www.luisllamas.es/usar-arduino-con-los-imu-de-9dof-mpu-9150-y-mpu-9250/

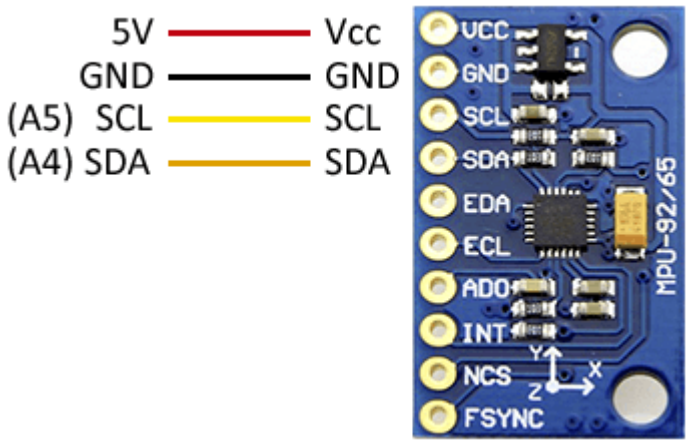


Ilustración 16 Tarje IMU. www.luisllamas.es

Hay que destacar que no todas las placas basadas en el mimo chip son iguales. La nuestra es una placa azul, de fabricante no especificado. No tenemos referencias técnicas inequívocas sobre ese modelo de placa, aunque es frecuente ver sus fotos en la web, en ventas….

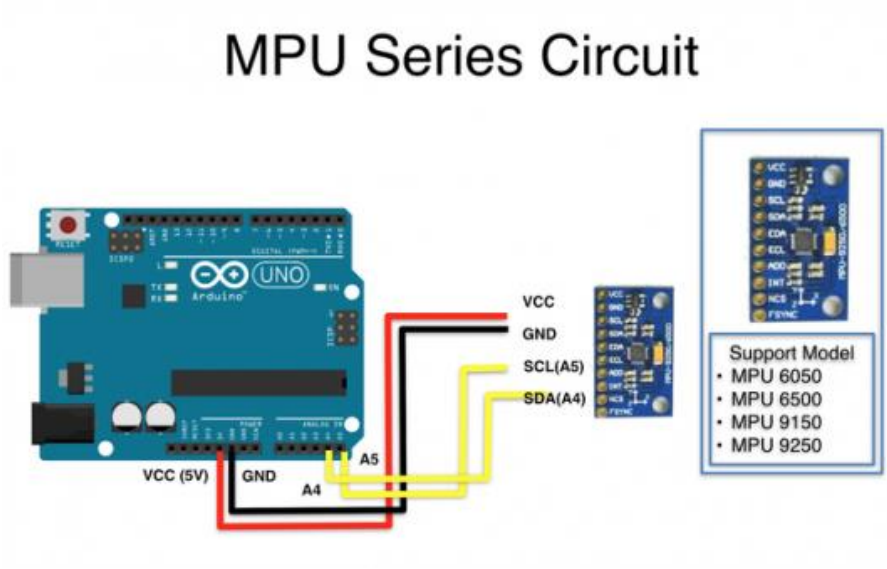


Ilustración 17 Conexión Arduino con MPU 9250

http://projectlab.co.th/2017/06/04/mpu-9250-hookup-guide/

https://www.instructables.com/id/MPU-91509250-IMU/

Finalmente se ha realizado la conexión de MPU9150 a un Arduino UNO, según una referencia en la que señalaban las siguientes conexiones (MPU-uno) : (SCL=A5, SDA = A4, V=3.3, Gnd=gnd). La librería que he usado está disponible on line desde el menú de gestión de librerías del propio IDE 1.8.5. Es una librería para el MPU 6050 , en la versión v1.4, y al correr sus ejemplos funciona. El 6050 está incorporado en el 9150, pero solo aporta la parte de aceleraciones y giros, no la parte de magnetómetro.

Los ejes del plano de la placa son x-y, y si ese plano es horizontal, la deriva en ángulos en los ejes x, y es muy pequeña, y con media =0. El único eje que tiene una deriva de giro poco estabilizada es el que en cada momento esté vertical, perpendicular al plano horizontal. Si se cambia la posición de la placa,

la inestabilidad en la deriva de giro del eje vertical se traslada al que en ese momento esté ocupando esa posición.

El magnetómetro que incorporan los chips 9150 y 9250 lo usa el propio chip para fusionar con los giróscopos y eliminar así la deriva de los mismos. Con ello los ángulos de navegación quedan libres de deriva temporal: escora, trimado y rumbo pueden ser usados directamente de la salida del chip.

El chip 9250 es de segunda generación respecto al 9150, y es más estable y preciso.

El ejemplo tiene una rutina inicial de calibrado que requiere unos 3 segundo con la placa inmóvil.

Después, manda por el puerto serie a la pantalla del PC una colección de datos cada periodo (configurable, ahora está a 1000 ms). Los datos son, en cada uno de los 3 ejes (x,y,z):

- Aceleración lineal = dV/dt
- Velocidad angular o dAng/dt

Esos son los 2x3= 6DOF que mide el sensor.

Además, da un dato de temperatura, unos 35ºC, posiblemente la del propio chip del módulo. Puede servir para monitorizar el estado de refrigeración de la misma.

Adicionalmente, como datos derivados (creados con el módulo de fusión del chip), el ejemplo da también

- Angulos de pose o actitud:
    - Angulo de trimado
    - Angulo de escora
    - Angulo de rumbo azimutal

Y podría dar velocidades lineales y posición, a partir de integrar:

- V = integ (aceleración, dt)
- Espacio = integ (Velocidad, dt)

NOTA:

El módulo hay que alimentarlo a V=3.3, porque así es la especificación.

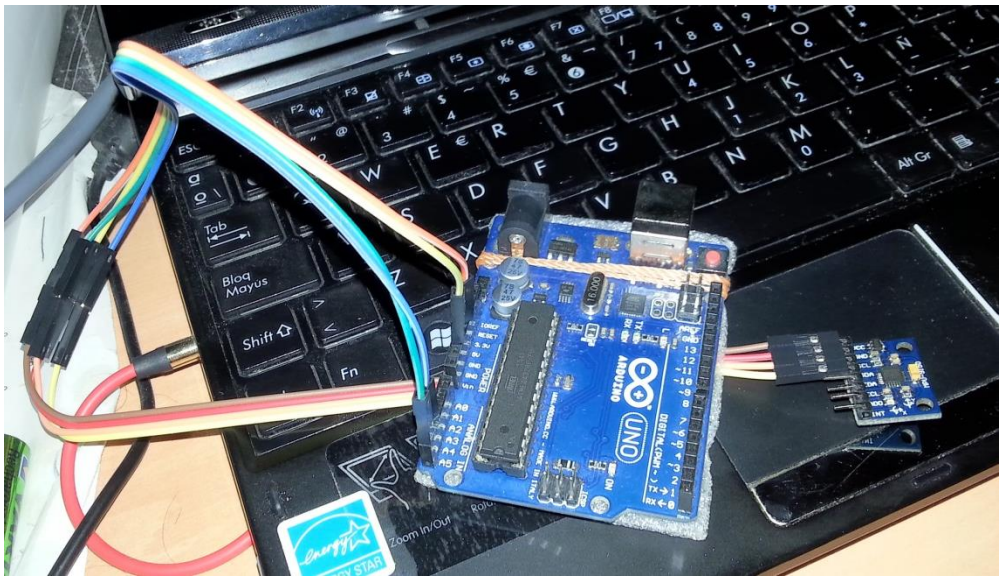


Ilustración 18 Conexión placa Arduino con tarjeta MPU9250. Fuente propia.

Posteriormente (19-8-2018) también he probado un MPU9250, con diversas librerías de ejemplo, en una placa Arduino MEGA, alimentando a 3.3V, funcionando perfectamente.

Se comprueba que los ángulos de actitud son un resultado de cálculo a partir de los diversos parámetros medidos por la IMU, y que necesitan algún tipo de filtro suavizador, para reducir su 'ruido'.

También se ha probado con un adaptador de niveles de tensión lógicos entre 5 y 3.3V, bidireccional, de 4 canales. El resultado ha sido el mismo. Se concluye que la placa 9250 es tolerante a la lógica de 5V del Arduino Mega, cuando se alimenta a la placa 9250 con 3.3V directamente, y por tanto no se requiere de adaptador de niveles lógicos externo.

12.5 Actuadores. Regleta de relés

Para controlar las conexiones de los circuitos eléctricos y gestionarlas necesitaríamos alguna regleta de relés, adecuada para controlar por el Arduino.

https://www.luisllamas.es/arduino-rele-estado-solido-ssr/

Sin embargo, parece que existen algunas limitaciones en los de estado sólido disponibles para comprar vía web, y los convencionales.

En particular, los de estado sólido cuestan el doble que los mecánicos y son menos abundantes.

La configuración NO/NC (Normally Open, Normally Closed, el circuito eléctrico) no es configurable, y es siempre NO. La elección entre trigguer on High/Low (H/L) no es configurable.

La operación deseada para el sistema de relés será la siguiente:

Si hay un fallo en la energía de la lógica, que el sistema de la carga (de fuerza) no arranque, y si estaba arrancado que se pare. Es decir, deseamos que haga trigger on H, que esté normalmente en L, y que si cae (accidentalmente) de H a L, el sistema de la carga se pare (a lo sumo se pretendería que en esa situación pase a una posición de salvamento o vuelta a casa).

Con todas estas posibilidades abiertas, optaremos por unos relés convencionales (no de estado sólido), ya que son mucho más flexibles en las configuraciones posibles.

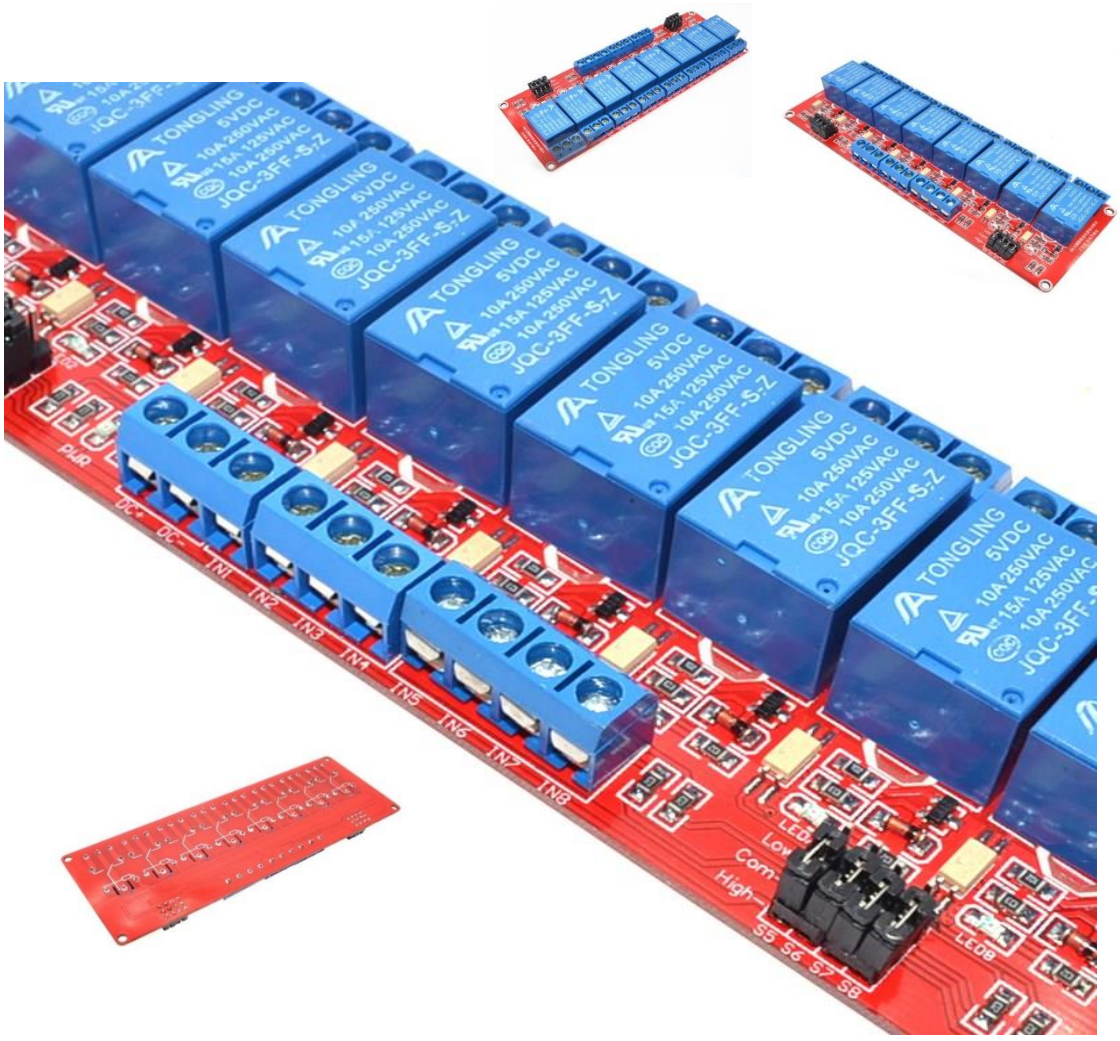


Ilustración 19 Regleta de relés. Ebay.es

12.6 Sensores para Medida del Consumo de corriente y tensión.

Una vez estructurada la red de circuitos, se puede analizar el consumo de cada uno de ellos, para conocer su estado de funcionamiento o fallos por sobrecarga.

Así se puede medir en varios puntos, bornes principales de batería, en cada batería, bornes de alimentación al motor principal, y a cada uno de los principales consumidores.

Además, relacionado con el control de los DCmotor, sería útil conocer cuál es la tensión real suministrada en cada momento al motor, la intensidad que está consumiendo, y con ellos estimar par y rpm. Para ello también ayudaría tener un encoder capaz de medir la velocidad real del motor. Si eso se consigue, a partir de las curvas características del motor se obtiene el par real mecánico y las rpm.

- Par = función (tensión, intensidad, rpm)

Hay que ver cómo implementar sensores de tensión e intensidad con arduino. Existen placas baratas para ello.

http://henrysbench.capnfatz.com/henrys-bench/arduino-projects-tips-and-more/arduino-max471-power-meter-tutorial/

Los hay que solo miden corriente, o que miden tensión y corriente. El que me interesa es el de ambos. Creo que debe tener 2+2 terminales de conexión, y lo hay en ebay.

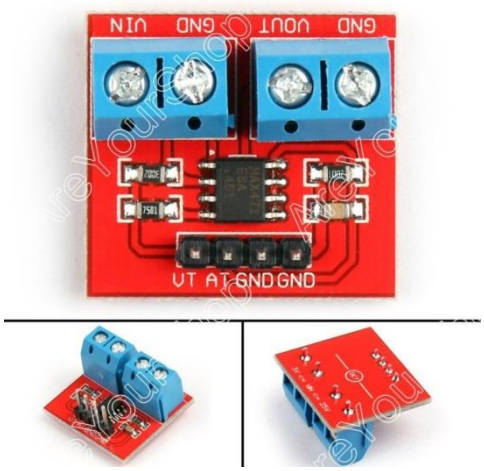


Ilustración 20 Sensor de consumo. Sr Ferrete.

- https://www.ebay.com/sch/i.html?LH_CAds=&_ex_kw=&_fpos=&_fspt=1&_mPrRngCbx=1&_nkw=MAX471+arduino+voltage+current&_sacat=&_sadis=&_sop=12&_udhi=&_udlo=&_fosrp=1

- https://www.luisllamas.es/arduino-intensidad-consumo-electrico-acs712/

- https://www.luisllamas.es/arduino-sensor-corriente-sct-013/

- https://programarfacil.com/blog/arduino-blog/sct-013-consumo-electrico-arduino/

- https://uge-one.com/max471-dc-voltage-and-current-sensor-module-2-in-1.html

- http://hobbycomponents.com/sensors/900-max471-gy-471-3a-current-sensor-module

- https://www.didacticaselectronicas.com/index.php/sensores/sensor-de-corriente-de-alta-precisi%C3%B3n-max471-detail

- https://www.ebay.com/p/1pcs-Max471-Voltage-Current-Voltage-Amplifier-Test-Sensor-Module-for-Arduino-

USA/578981602?iid=282782587274&_trksid=p2047675.m4097.l9055

- https://i.ebayimg.com/images/g/1KIAAOSw4CFYrd5A/s-l1600.jpg

12.7 Sensor de humedad y temperatura

También quiero un control de humedad y temperatura dentro del espacio, para conocer el valorar condensaciones, etc.

Finalmente he conseguido uno que puede alimentarse tanto a 3.3V, como a 5V, y tiene medidas de Presión atmosférica, temperatura y humedad relativa. Su consumo es muy bajo.

Es una versión del GY68 BMP180.

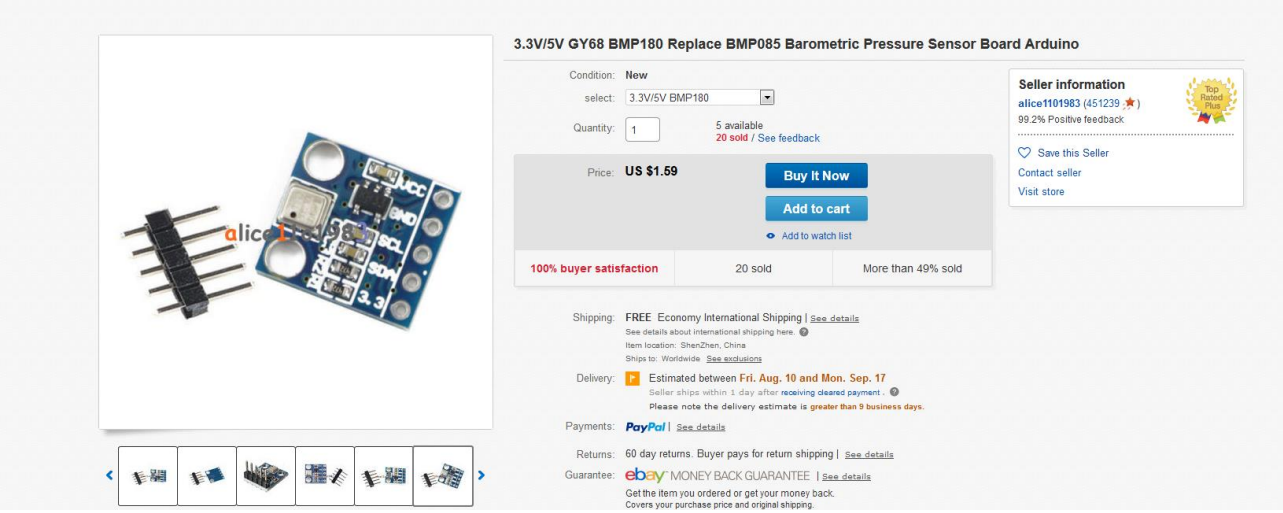


Ilustración 21 Sensor de temperatura, humedad y presion admoferica, para Arduino. Ebay.es.

12.8 Refrigeración por aire circulante.

Necesitaremos también un ventilador para homogeneizar la temperatura del aire, que se ha de usar como refrigerante de la electrónica, (quizá con un arranque intermitente), según algún parámetro de temperatura o humedad próximo a la electrónica de potencia y control.

El aire se puede hacer circular a lo largo de la pared de un costado del AUV, hasta llegar a la tapa del extremo, donde pasará a volver por el otro costado hasta la otra tapa opuesta, y cierra el recorrido volviendo al punto inicial, donde está el ventilador. Para que la circulación del aire sea de ese modo habrá que poner algún tipo de mamparo longitudinal (puede ser una lámina de plástico transparente, por requisitos didácticos).

Los costados del tubo cilíndrico de AUV están por fuera en contacto con el agua, cuya temperatura se asume <20ºC para nuestros propósitos. De ese modo, el aire del interior, en contacto con esa pared deberá acondicionar su temperatura, cediendo el calor que recoge de los motores y la electrónica.

Estimamos que la potencia térmica a disipar es relativamente pequeña. Puntas de 4 W durante 10 segundos, y largos periodos con 1 W (5V*200 mA), por lo que no deberíamos necesitar una refrigeración más sofisticada.

Podríamos estimar la capacidad refrigerante a través de la superficie mojada del AUV con algunos datos básicos:

- Superficie de intercambio aire-agua
- Temperaturas del aire interior, y del agua exterior,
- Potencia disipada en el interior
- Conductividad térmica del metacrilato (MC) (sacada de http://www.plasticos-mecanizables.com/plasticos_metacrilato.html)

Conductibilidad térmica a 20º	DIN 52612	0,19	W/(m*K)

Y con todo ello estimar temperatura de equilibrio, y tiempo en que se alcanza. Solo estimando la temperatura de equilibrio (régimen estacionario de flujo de calor) podemos obtener:

- Pot = k. A/L * deltaT
- dT = 1.1ºC para 2W, considerando solo intercambio a través de la superficie lateral del cilindro 600x100x4 de metacrilato.

En la práctica, el flujo de calor se realiza en forma escalonada. De la electrónica al aire de refrigeración interior, de este a la masa de aire general interior, y de ella al agua del exterior. El principal problema está en hacer eficaz la transmisión de calor entre la electrónica y la masa general de aire interior. Ahí es donde hay que facilitar el flujo de aire, evitando zonas de remanso alrededor de los puntos calientes.

A efectos prácticos usaremos un ventilador centrifugo que creara suficiente turbulencia de aire, como para crear el intercambio de calor sin guiar el flujo de aire.

Ilustración 22 Ventilador centrifugo. Ebay.es

12.9 Gestión de energía

Todos los sistemas que constituyen el ‘cerebro’ y las ‘manos’ del control del AUV, se alimentan con corriente continua a tensiones en el rango 5V-12 V. La energía eléctrica requerida debe disponerse en unas baterías con suficiente capacidad para las misiones previstas. Las baterías podrán ser recargadas periódicamente, a partir de una fuente de alimentación adecuada disponible ‘en tierra’.

Las misiones de navegación del AUV se realizan fuera del control normal de las personas. Por ello, y dado lo inaccesible de su entorno, tendremos que contemplar situaciones ‘anómalas’ en las cuales debería ser posible (y de modo automático) recuperar el control, abortar la misión, vuelta a casa, y rescatar el AUV.

Todo esto nos va llevando a diseñar una planta eléctrica bien estructurada, con cierta similitud a la de las plantas reales de los buques. La causa de que se parezcan es que en ambos casos se busca fiabilidad, seguridad ante imprevistos, y recursos para superar incidentes.

La gestión de la energía es importante. La seguridad va a depender de un buen diseño de ese sistema.

Un primer paso puede ser diferenciar dos fuentes de energía, para asegurar una cierta redundancia.

Igualmente podemos segregar los consumidores en dos bloques: los circuitos de fuerza, (que requieren bastante consumo, aunque tienen periodos relativamente cortos de operación), y los circuitos de control (que deben funcionar todo el tiempo, pero con bajo consumo). Así se pueden configurar como 2 barras separadas (aunque interconectables).

En principio cada una de las dos barras tendrá una fuente distinta, aunque en

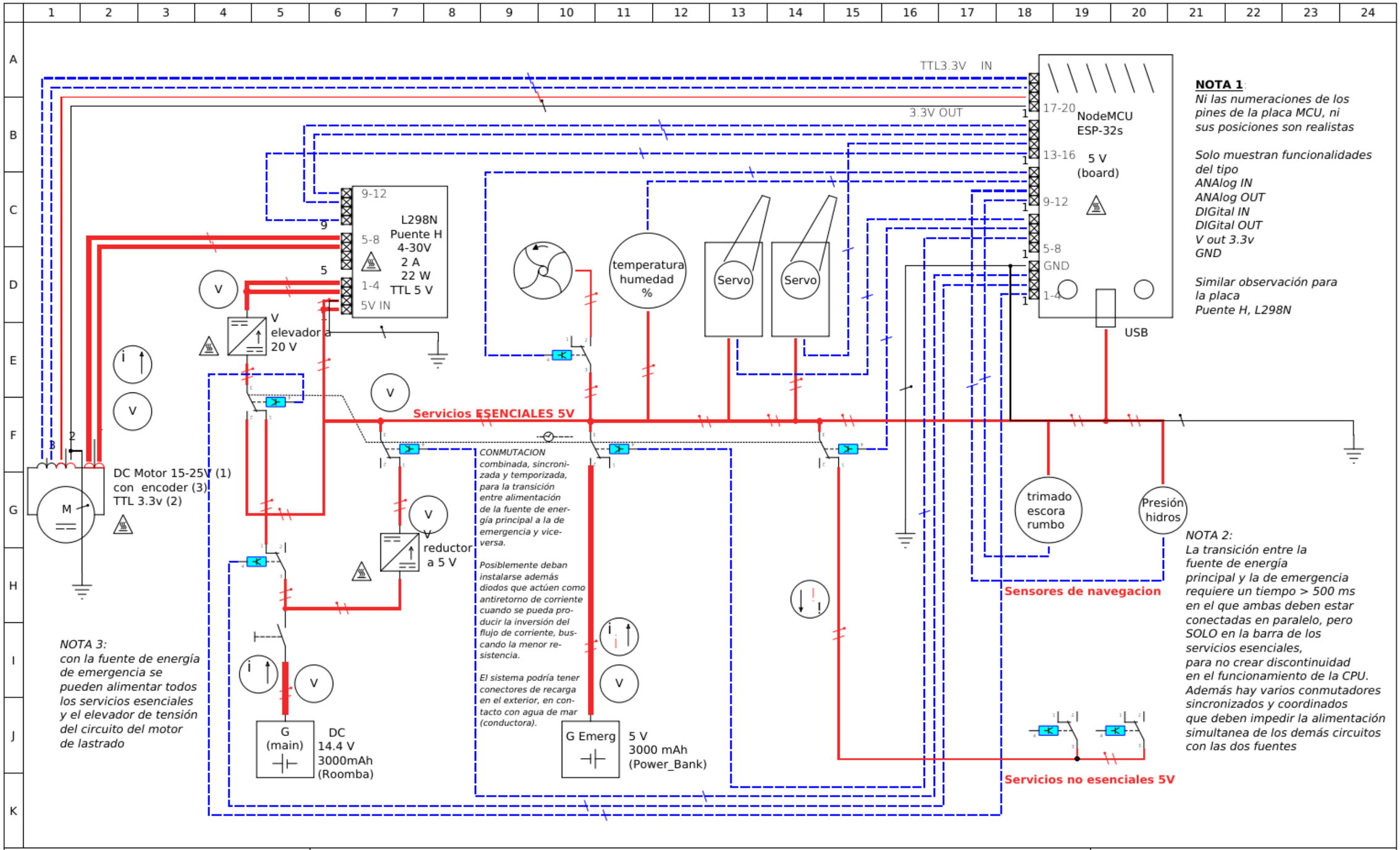
caso de necesidad, ambos circuitos se pueden poner en barra común, y ser alimentados por una cualquiera de las fuentes, o por las dos en paralelo. También puede ser deseable que periódicamente se puedan permutar las dos fuentes entre las dos barras, para que ambas envejezcan de modo similar. Deseamos conocer tanto la reserva en la fuente principal y en la secundaria, como el consumo en cada momento de todos y cada uno de los elementos relevantes.

Si distribuimos la energía en 2 fuentes de la misma tensión, deberíamos tener la posibilidad de conmutar mediante relés a varios posibles estados:

1- On/off Conectada fuente1 a barra A,
2- On/off Conectada fuente2 a barra B
3- On/off Conectada la permuta de las fuentes A y B
4- On/off Conectada barra A a barra B
5- On/off Conectada carga1 a barra A,
6- On/off Conectada carga2 a barra A,
7- On/off Conectada carga1 a barra B,
8- On/off Conectada carga2 a barra B,

12.10 Esquema unifilar de la instalación eléctrica.

Resumiendo todas las necesidades y circunstancias descritas anteriormente, hemos establecido que nuestra planta eléctrica sea la del esquema de la página siguiente. En él se destaca que la fuente de energía principal es una batería estándar de robot de limpieza Roomba de 14 voltios y la fuente de emergencias es una powerbank 5v por 3000 ma/h.



Autor: David Espinosa García	Sistema eléctrico del AUV. Diagrama unifilar-mixto	Archivo: AUV_sistema_electrico
Fecha: 29/8/20	Creado con QelectroTech, https://qelectrotech.org/:	Folio: 1/2

Ilustración 23 Esquema unifilar de la instalación eléctrica. Fuente propia.

Cada interruptor se puede identificar con un relé, que nos permitirá su gestión. Además, en puntos adecuados dispondremos sensores de Tensión e Intensidad de corriente, de modo que podamos monitorizar en cualquier momento la forma en la que esté trabajando la instalación. Eso ayudará a tomar decisiones acertadas ante las incidencias, incluyendo procedimientos automáticos de protección tanto de cada uno de los elementos del AUV, como de todo el AUV en su conjunto.

13. Las comunicaciones con el Arduino vía Bluetooth

Las comunicaciones que pretendemos son entre el sistema dentro del AUV, y un mando en el exterior. No deseamos limitarnos a un cable que atraviese el casco por motivos de fiabilidad de la estanqueidad, estructura, etc. No obstante, podría ser una opción primitiva con algunas ventajas para el rescate, etc.

En su lugar vamos intentar una comunicación inalámbrica. Entre las tecnologías disponibles, el protocolo Bluetooth tiene gran implantación, y es asequible técnica y económicamente. Su principal limitación es el corto alcance (unos 5-10 m en el aire), y la muy baja penetración en el agua (apenas 1 cm), debido a su frecuencia de 2.4 Ghz.

Ese alcance limitado solo nos va a permitir evitar tener que abrir el AUV para la programación o modificación del software de su sistema. No obstante, consideramos suficiente poder pasar de ese modo instrucciones al AUV cuando esté en superficie, y a nuestro alcance. Asumiremos que podemos programar cada misión para que termine con éxito en la superficie, y que podamos recuperar el AUV sin mayor dificultad.

Hemos renunciado a que la comunicación inalámbrica pueda atravesar el agua, a la profundidad operativa del AUV, porque sabemos que en los sumergibles reales existen los mismos problemas, y se acude a hidrófonos, etc, que son tecnologías que escapan de nuestras capacidades e intención.

Los fundamentos del Bluetooth
https://aprendiendoarduino.wordpress.com/tag/hc-05/

Los Smartphone incorporan BT, así como una gran variedad de APP que pueden adaptarse para gestionar señales y canales para controlar dispositivos al otro lado de la comunicación BT.

13.1 Interacción SmartPhone Android-Arduino

Para comunicarnos con el Arduino vamos a usar un dispositivo Android, por su amplia disponibilidad. En el Android tendremos que preparar un programa de comunicaciones que nos facilite indicarle al Arduino las instrucciones para las misiones, la navegación, etc.

Un elemento muy importante para la interacción es el interface gráfico de usuario (GUI) que aparece en la pantalla del Android. Deseamos que el control sea de tipo táctil, con pocas entradas vía teclado.

Para manejar una GUI adecuada a nuestro proyecto disponemos de 2 opciones: encontrar una que exista y nos resulte adecuada, o preparar nuestra propia GUI. Después de hacer una barrido, no encontramos ninguna GUI ya configurada que cumpla nuestro interés, aunque existen otras para otro tipo de robots:

Por ello intentaremos hacer una GUI adecuada a nuestro AUV. A su vez, para crear GUIs hay dos grandes grupos de aplicaciones disponibles a través de Google Play, para la plataforma Android:

1- Las que tienen una distribución ya definida de botones, reguladores, etc, y están listos para ser usados ya que mandan directamente códigos via BT o incluso Wifi al receptor BT asociado a un canal de comunicación serie de la tarjeta Arduino. pueden trabajar offline de internet

2- Los que admiten que el usuario configure los botones, deslizadores, etc a su medida, con muchas posibilidades y permitiendo una interfaz de usuario GUI muy atractiva y adaptada al gusto de cada uno.

Esta últimas suelen tener el inconveniente de que el código configurado por

el usuario solo está disponible en la nube del fabricante de la App, de modo que puede usarse SOLO con conexión a intenet, online.

Por el contrario, las primeras son autónomas para el usuario.

Aunque entre la segunda hay alguna muy atractivas, como Blynk o XY controles, no cumplen nuestra especificación de autonomía, y las debemos descartar.

Restringiendo la selección a las autónomas, que trabajen offline, tenemos:

1- ArduinoDroid, con IDE que permite la programación desde el propio Android, via USB OTG . No veo ventajas respecto al IDE, salvo la autonomía. No permite hacer GUIs para interactuar con el Arduino.

2- Roboremo es otra aplicación que permite crear un GUI en el Android, y cada uno de los elementos del GUI puede hacer alguna cosa, mandar o recibir info o datos hacia el Arduino.

   a. https://www.roboremo.com/uploads/2/4/5/7/24571986/manual_v200.pdf

3- Bluetooth Electronics, from KEUWLSOFT

   a. http://www.keuwl.com/apps/bluetoothelectronics/
   b. PARECE CONFIGURABLE OFFLINE Y ADECUADO.

Usar ardudroid para controlar los pines de Arduino desde el móvil:

1- App: https://play.google.com/store/apps/details?id=com.techbitar.android.Andruino&hl=es_419

2- Manual de uso: http://www.techbitar.com/ardudroid-simple-bluetooth-control-for-arduino-and-android.html

3- Código en Arduino: https://github.com/jecrespo/Aprendiendo-Arduino/tree/master/Ejercicio41-Bluetooth/ardudroid

Otro tipo de configuración permitiría usar 2 móviles conectados a la misma placa Arduino, de modo que el Arduino interactúe con cada uno de los 2 móviles. Uno de ellos puede usarse como terminal de usuario para dar instrucciones al Arduino, o recibir información de él. El otro móvil (instalado en el interior del AUV) puede usarse como sensores ofrecidos al Arduino via BT, para que éste (el Arduino) gestione esos sensores, o reenvíe esa información al otro móvil del usuario (en el exterior del AUV). Una forma de hacerlo es con la app sensoduino.

Mandar a Arduino los datos de los sensores del móvil con sensoduino
http://www.techbitar.com/sensoduino.html

1- App: https://play.google.com/store/apps/details?id=com.techbitar.android.sensoduino

2- Manual de uso: http://www.techbitar.com/sensoduino.html

Eso y más se explica en este enlace, con muy buenas indicaciones sobre todo esto:

https://aprendiendoarduino.wordpress.com/2016/11/13/bluetooth-en-arduino/

Existen en el mercado DIY una gran diversidad de variantes de módulo BT para Arduino.

http://www.martyncurrey.com/bluetooth-modules/

El que nosotros tenemos, y vamos a usar, es el HC-05 – zs-040, que pasamos a describir.

http://www.martyncurrey.com/hc-05-and-hc-06-zs-040-bluetooth-modules-first-look/

Arduino with HC-05 (ZS-040) Bluetooth module – AT MODE

https://www.gme.cz/data/attachments/dsh.772-148.2.pdf

Nuestro módulo es, en particular:

- Placa zs 040,
- Variante HC-05,

Admite más funciones de configuración que el HC-06, aunque para ello tiene algo más de complejidad, 6 pines, botón paso a AT-mode,

El siguiente código y montaje permite interactuar desde un terminal Android (un smartphone común) con el Arduino, via BT

http://www.martyncurrey.com/using-an-arduino-mega-with-a-hc-05-zs-040-at-mode/

http://www.martyncurrey.com/using-an-arduino-mega-with-a-hc-05-zs-040-at-mode/#comment-3954

Utilizar el módulo de Bluetooth requiere el uso de un puerto serie de nuestra placa Arduino. Por tanto, mientras usemos el módulo de Bluetooth no podremos usar el puerto serie incluido en el hardware en las placas modelo Uno, Mini, y Nano, ya que solo existe un puerto. En el modelo Mega no tiene este problema, ya que incorpora 4 puertos serie Hardware.

Mientras estemos cargando un nuevo programa en la placa Arduino tenemos que desconectar el módulo Bluetooth, dado que la programación se realiza a través del puerto serie.

Si realmente necesitásemos varios canales simultáneos de comunicación serie, podríamos emplear la librería SoftSerial para establecer una comunicación de puerto serie por cualquier pareja de pins digitales, aunque ello supondrá un coste adicional de tiempo de proceso en Arduino.

La conexión es sencilla. Alimentamos mediante Vcc (+5V) y GND. Posteriormente conectamos el TXD (pin de transmisión) y RXD (pin de recepción -CUIDADO AQUÍ), a los opuestos de la placa Arduino (cada TXD a un RXD).

La Advertencia previa es que, como los pines de comunicaciones del HC-05 son de nivel 3.3V, hay que intercalar un divisor de tensión que adapte-rebaje a 3.3V la salida de 5V del Arduino). Así quedarían las conexiones del módulo, con los pines de Arduino.

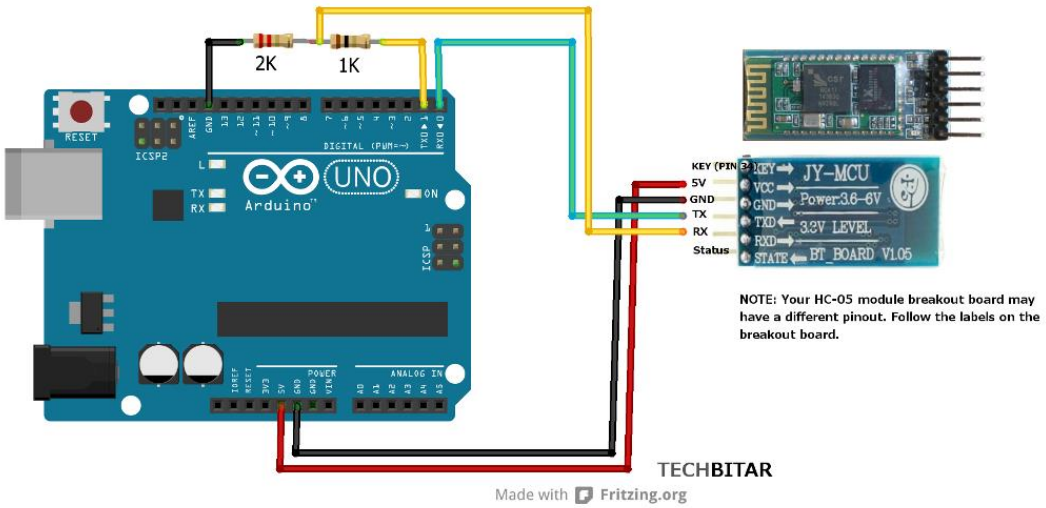


Ilustración 24 Conexión Arduino con coneccion bluetooth . Sr. Ferrete.

Información de la tarjeta.

The breakout board has a 3.3V regulator which can convert Arduino's incoming 5V power but you still have to drop the Arduino's 5V pins down to 3.3V. The simplest way is a voltage divider from two resistors but you can use a level shifter such as the CD4050 level shifter IC.

I use 2K & 1K resistors as voltage divider to drop the Arduino's 5V TX pin to 3.3V, which is the operating voltage of the HC-05 pins. You have to do this to protect the HC-05 3.3V RX pin. On the other hand, the Arduino 5V RX pin can handle the 3.3V sent from the HC-05 TX pin. Even if the HC-05 worked without the voltage divider, there's no telling for how long.

TO UPLOAD A SKETCH TO ARDUINO YOU MAY HAVE TO REMOVE THE POWER FROM THE HC-05 MODULE TO AVOID ANY CONFLICT OVER SERIAL. Alternatively, you can wire the HC-05's TX/RX to different Arduino pins supported by Arduino's SoftwareSerial library.

14. Diseño del menú de gestión del AUV.

Vamos a crear un conjunto de rutinas para el Arduino que hagan operaciones típicas necesarias en el control del AUV.

La gestión de las rutinas del AUV la haremos por medio de un móvil Android, a partir de una aplicación de menús gráficos con mando táctil, de tipo estándar.

Vamos a establecer el catálogo y funcionalidad deseable de esas rutinas.

1- Establecer los parámetros operativos para la misión:
   a. El tiempo máximo de una misión, TMAX, transcurrido el cual, el AUV pasará a flotabilidad máxima y subirá a la superficie.
   b. pMAX
   c. RUMBO
   d. Vmin, tensión por debajo de la cual se debe liberar el lastre fijo

2- Autochequeo de comunicaciones y básico.

3- Muestra de todos los parámetros eléctricos relevantes: Tensión, intensidades en cada uno de los puntos relevante. Estado de los relés.

4- Muestra en pantalla de las Lecturas de todos los sensores.

5- Hacer el calibrado inicial de flotabilidad, previo a una misión. Trimado =0, Flotabilidad =0

6- Emprender una misión del tipo NAA a una inmersión máxima pMAX, y rumbo RUMBO fijo (requiere que primero se haya hecho el calibrado de inicio de misión)

7- Emprender una misión del tipo NAA en espiral, con diámetro estimado

DIAM

8- Disparo del lastre solido externo

9- Paso al modo rescate

10- Paso al modo emergencia

Estos datos a través de un menú simple creado en una aplicación para el teléfono móvil, el cual nos va a dejar hacer movimientos de prueba desde el teléfono, pudiendo hacer el movimiento de la masa móvil y el cilindro de inundación del submarino.

En la siguiente página se muestran imágenes de la electrónica del submarino montada, ya que el trabajo no permite videos se deja imágenes del equipo terminado. Los movimientos se pondrán a disposición de comité de evaluación para que puedan ver su funcionamiento.

Ilustración 25 Soporte para electrónica con ventilador y relés 1. Fuente propia

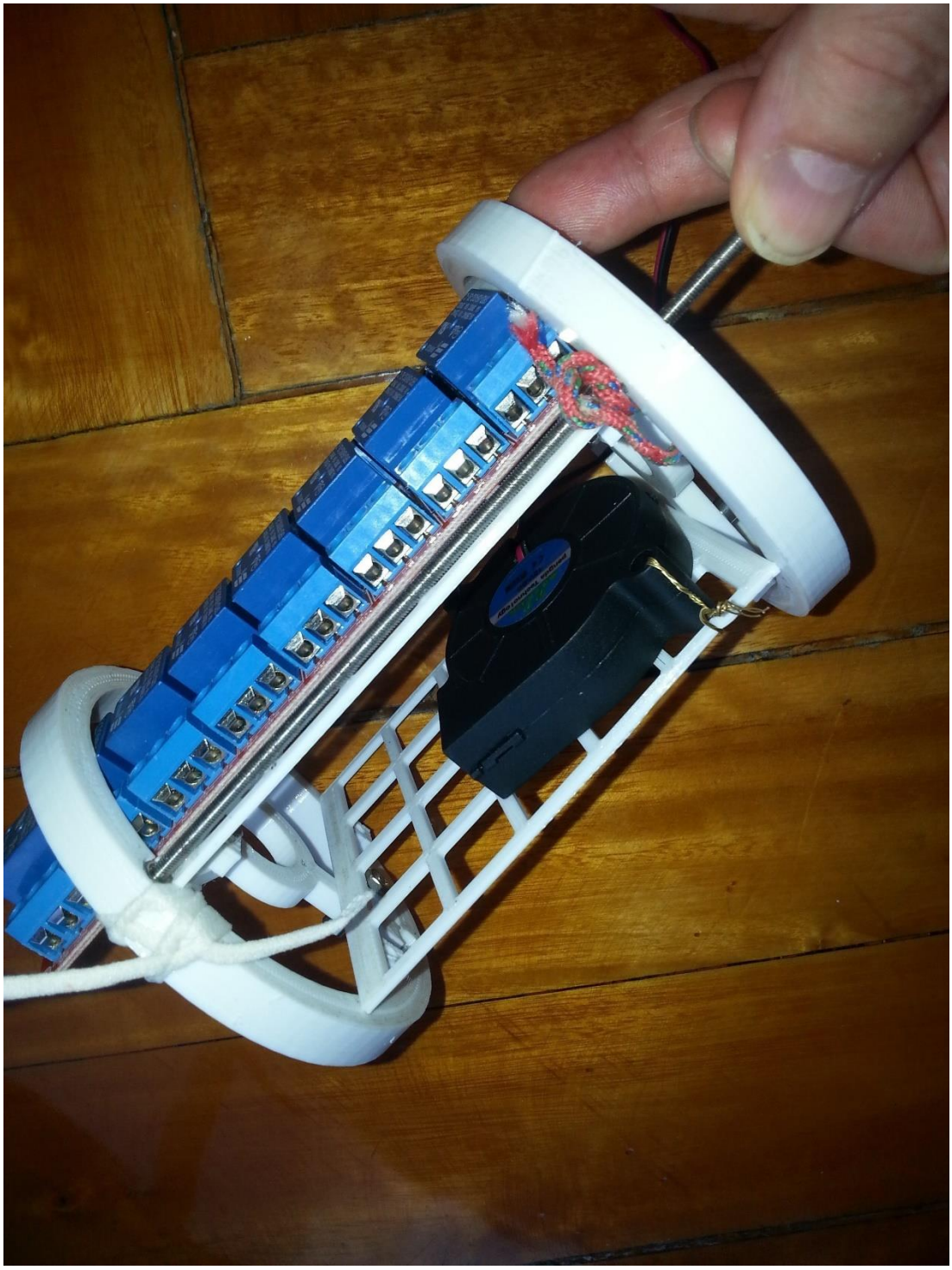


Ilustración 26 Soporte para electrónica con ventilador y relés 2. Fuente propia

15. Simulación y control de AUV GLIDER.

15.1 Necesidad y utilidad de la simulación.

En este apartado vamos a recoger de una forma más simple y breve la esencia del anexo denominado **ANEXO: Simulación y control del AUV GLIDER** Los detalles y explicaciones más completas pueden verse allí.

15.2 Modelo matemático del AUV GLIDER para las simulaciones

Mientras el AUV Glider no esté construido y disponible para pruebas, no tenemos la opción de conocer la realidad de su comportamiento. Algunos de los parámetros de diseño condicionan totalmente el comportamiento final del sistema. Por ello es muy importante tener un modelo numérico adecuado para las simulaciones, y que nos permita estimar los efectos de los cambios en los parámetros fundamentales.

En la bibliografía encontramos una referencia básica [**Graver 2005]** que está reiteradamente citada en todos los estudios relacionados con los AUV Glider. En la Web hemos obtenido la tesis de Gaver. Pero también hemos encontrado algo mucho más difícil hasta ahora. Hemos conseguido el código Matlab que implementa los cálculos matriciales y de ecuaciones diferenciales que se describe en la tesis. Hemos comprobado que el código da los mismos resultados con los datos del ejemplo de referencia, y asumimos que funciona correctamente.

Eso nos permite realizar nuestras propias simulaciones con ese código. Nosotros vamos a usar OCTAVE en lugar de MATLAB, ya que es Open Source, y eso nos gusta más. Sus prestaciones son comparables.

Los parámetros que hay que introducir en el modelo para que represente a nuestro Glider tampoco los conocemos, pero los hemos estimado por semejanza dimensional a partir de los de otro modelo (Slocum) conocido en la bibliografía y bien descrito para ese propósito.

En la siguiente figura se muestra (sin alas) una visión esquemática del modelo teórico genérico

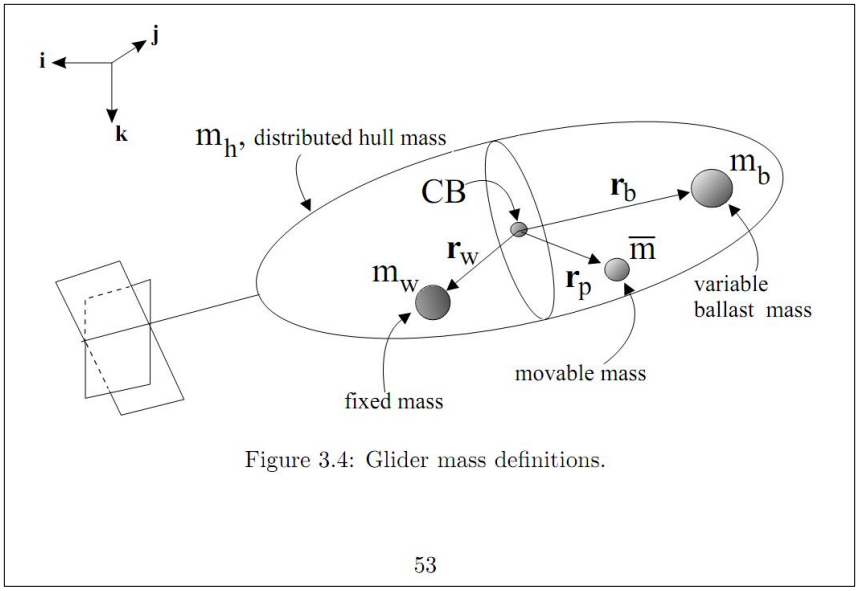


Ilustración 27 Modelo teórico geométrico GLIDR. Grave, 2005 Pagina 53

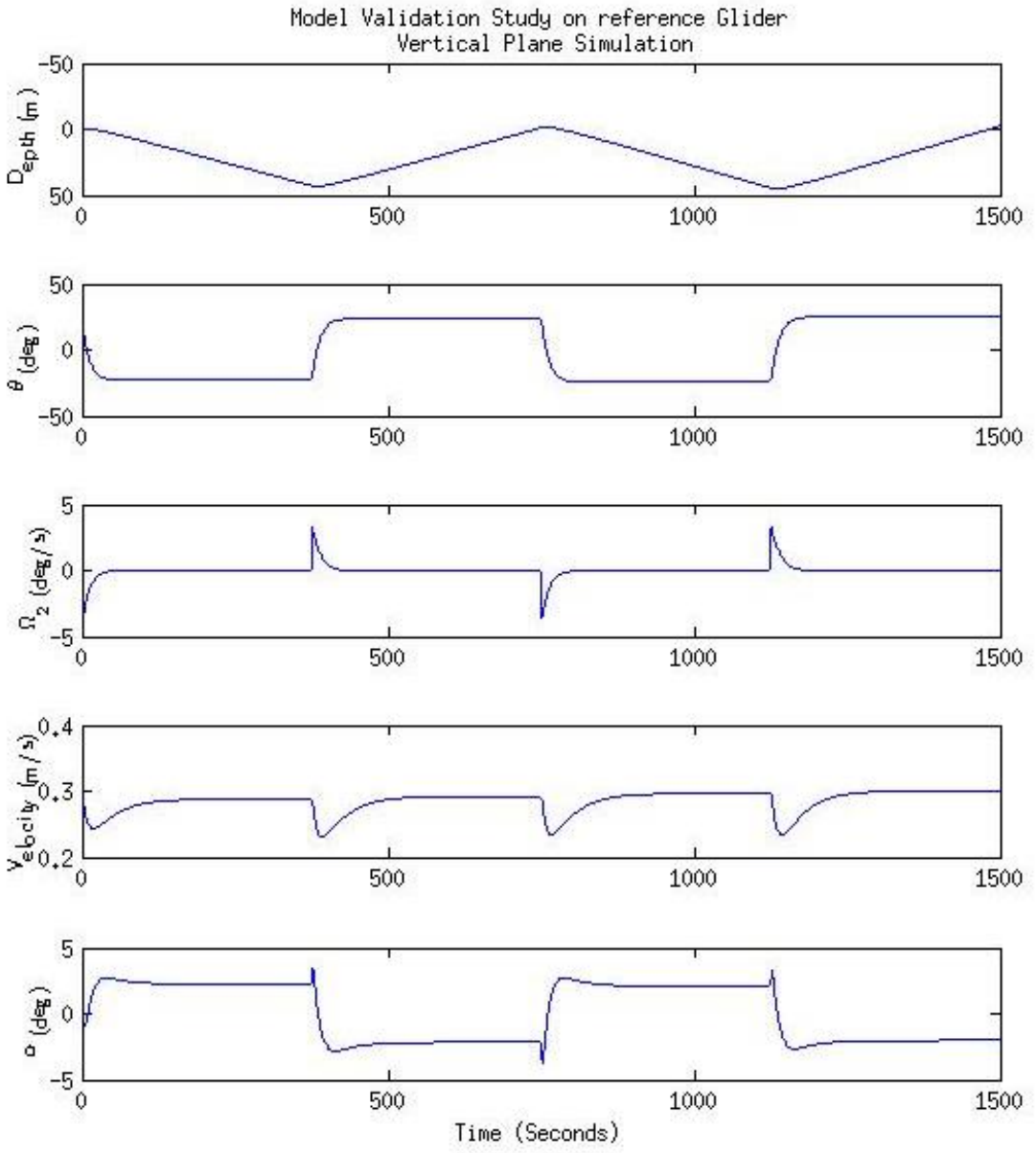
El control de la navegación se basa en dos actuadores interiores que cambian el estado de masas del Glider, alterando dos masas asumidas puntuales:

- Una masa, $\mathbf{m_b}$, de magnitud variable, pero de posición fija: el lastre (**b**allast en inglés). Comúnmente es líquido que se trasiega entre el interior y el exterior del Glider, y que modifica la flotabilidad del conjunto, dando una resultante vertical, hacia arriba o hacia abajo, que es la causa primera del movimiento del Glider. Esa es la forma de modificar la flotabilidad por el principio de Arquímedes. El mismo principio que también utiliza la naturaleza con la vejiga natatoria de los peces.
- Una masa móvil interior, $\overline{m}$,capaz de moverse longitudinal y transversalmente, y que con ello modifica la posición del Centro de Gravedad (CDG) del conjunto del Glider.

Después de haber estudiado en profundidad la tesis de **[Graver 2005],** y el código de [**Singh 2018]**, avanzamos con más seguridad en la simulación.

15.2.1 Prueba del modelo de simulación con OCTAVE.

La simulación del Slocum realizada en **[Graver 2005]** sirve de prueba de validación del código de **[Singh 2018]**. Son 4 planeos sucesivos a ± 30º de trayectoria, de 375 s de duración cada uno.



Esta figura, DE ELABORACIÓN PROPIA, es nuestra propia ejecución del code de **[Singh 2018]** con el ejemplo de datos del SLOCUM. Como dá los

mismos resultados que el ejemplo, entendemos que el code o codigo puede ser válido.

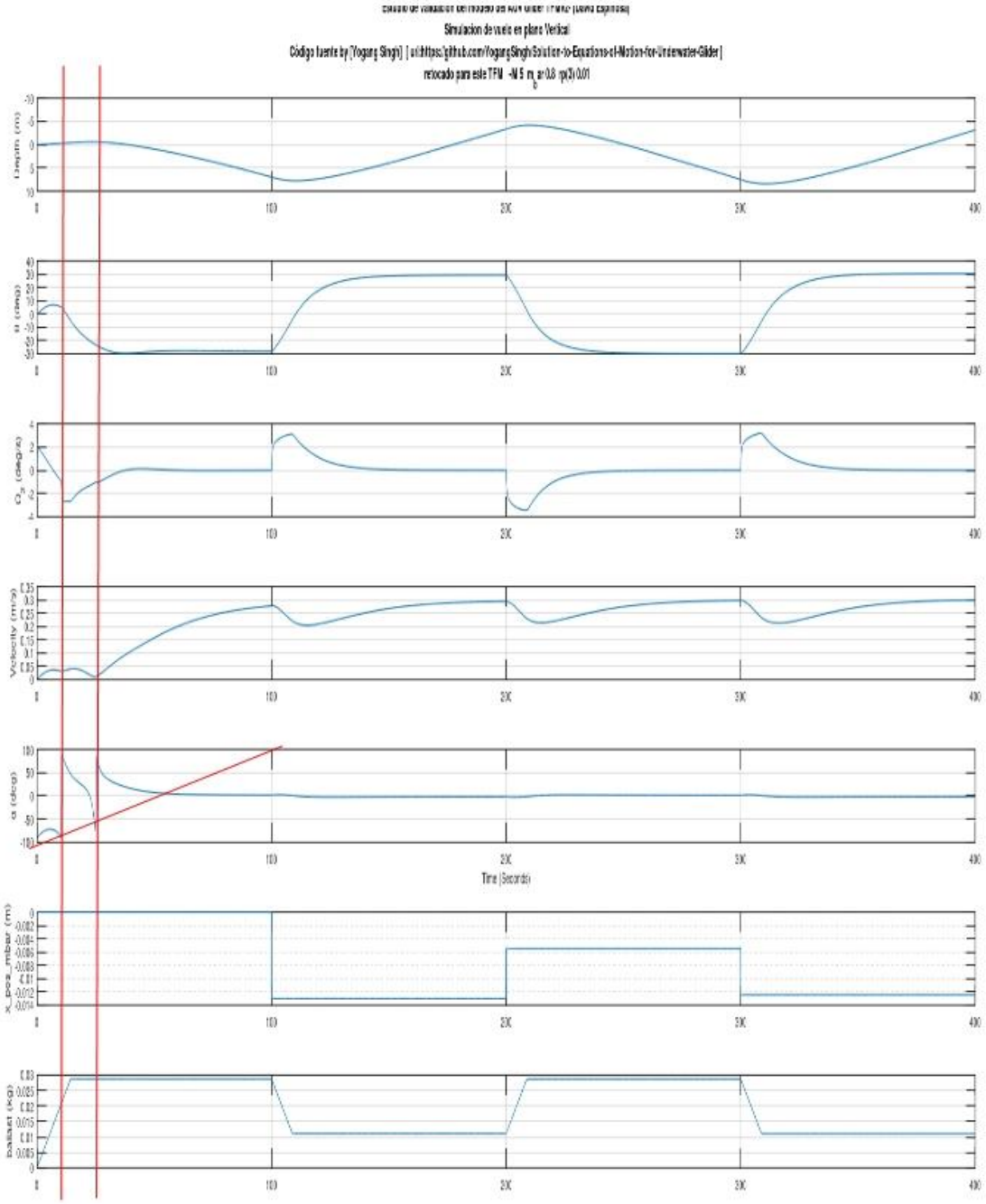
15.2.2 Determinación de los parámetros de nuestro AUV Glider para la simulación

Algunos de los datos de nuestro modelo podemos deducirlos directamente de su proyecto teórico, y otros, como los relacionados con la hidrodinámica, no. En el ANEXO se explica detalladamente cómo se consigue relacionar los parámetros de dos modelos con cierta semejanza geométrica a partir de sus masas. Eso nos ha permitido obtener unos parámetros para nuestro modelo.

15.2.3 Simulación de navegación. Comportamiento de nuestro Glider con los parámetros que hemos estimado.

En los nuevos parámetros estimados para nuestro AUV Glider hemos simulado varias carreras de Navegación Alternante Arriba y Abajo (NAA), cuyas salidas gráficas mostramos aquí. Las líneas representadas son la evolución a lo largo del tiempo, en abscisas (400s, = 4 x 100 s carreras), de los parámetros siguientes:

1. Profundidad,
2. Asiento (º),
3. Velocidad de cabeceo,
4. Velocidad lineal de traslación,
5. Alfa (ángulo de ataque º),
6. x_bar (abscisa o posición de la masa móvil),
7. Masa de lastre contenida en la cámara de flotabilidad de proa.



La navegación prevista era a una velocidad máxima V = 0.3 m/s, con un ángulo de planeo de 30º (en descenso y en ascenso), y no se ha fijado una profundidad máxima y mínima en la que hacer la inversión del movimiento, sino que esa inversión se ha mandado a hacer, según un temporizador, cada 100 s.ç.

La masa del Glider es de 5 kg, el parámetro de estabilidad es BG = 5 mm, la masa móvil es m_bar = 0.800 kg

15.2.3.1 Las maniobras sincronizadas y bien hechas

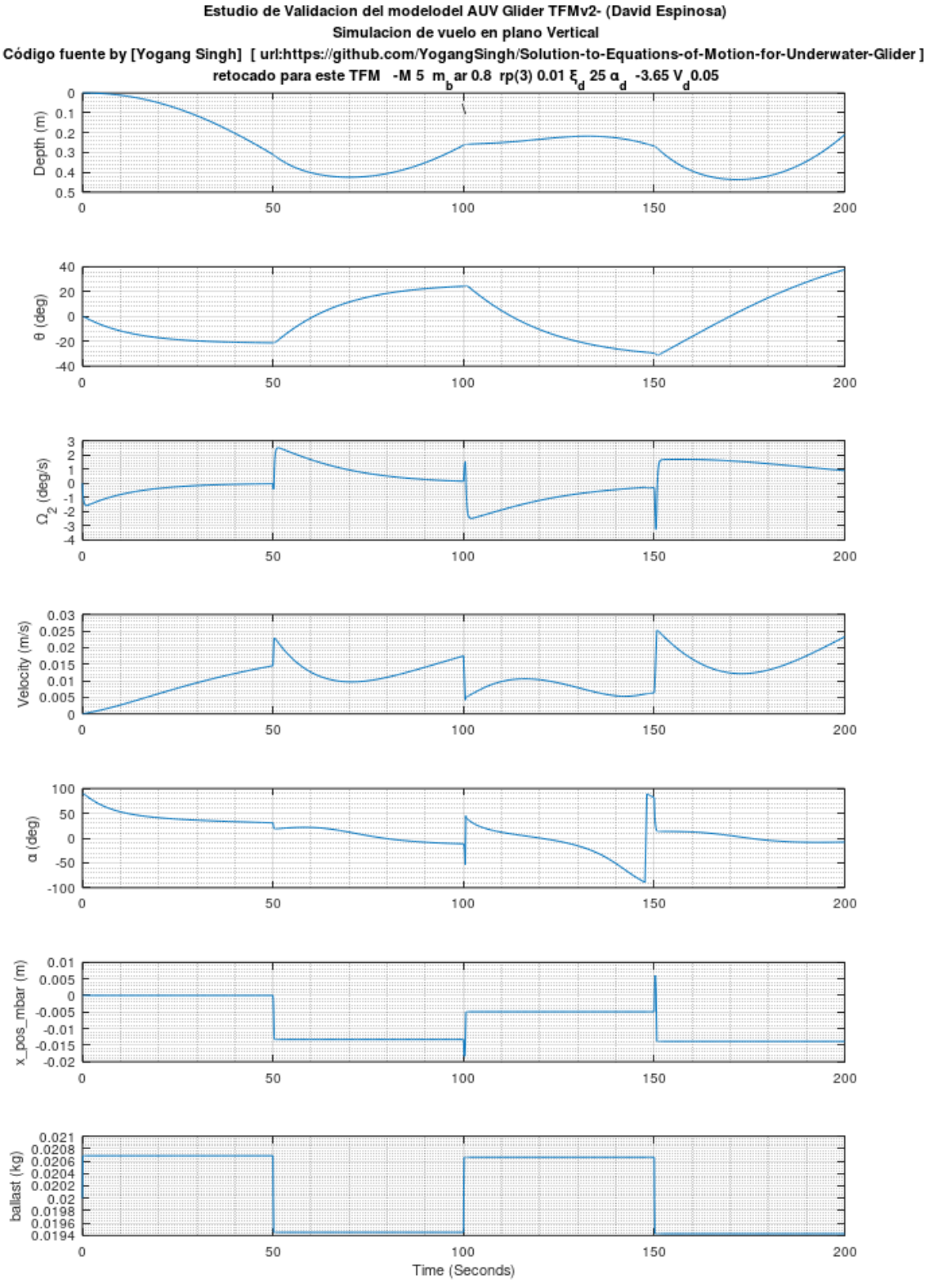
En la gráfica anterior se aprecia con mucha claridad que el inicio de la inmersión es accidentado, pero a los pocos segundos el AUV Glider coge velocidad, sustentación, y todo empieza fluir con suavidad.

El control, en nuestro caso, se reduce a mover a la velocidad adecuada (ni poca ni mucha) y en el momento adecuado, las masas de control (el lastre líquido variable y masa sólida móvil). Los estados finales de las masas de control son calculados a priori con unas fórmulas en las que conocemos todos los datos necesarios.

Una de las claves para que las maniobras se realicen bien, con transiciones suaves y con poca pérdida de energía, es que las posiciones de la masa móvil y cantidad de lastre se alcancen con precisión y en un tiempo relativamente pequeño, mientras la inercia de velocidad mantiene todo fluyendo correctamente. La calidad de la maniobra depende de la eficacia de las acciones de control, y los tiempos de respuesta dependerán de la fuerza de los motores y servos.

Si las gráficas de salida del simulador son líneas con saltos bruscos de valor o de pendiente, tales cambios abruptos nos señalan defectos en la ejecución de la maniobra, en las acciones de control: demasiado brusca, anticipada, tardía, mal sincronizadas, etc.

Mientras que la gráfica anterior representaba unas maniobras bastante correctas, la siguiente figura es un ejemplo de desatinos.



En este caso, una velocidad demasiado baja y cambios demasiado bruscos de las masas de control pueden afectar negativamente al flujo en el ala, y hacer que entre en pérdida con facilidad. El resultado es una mala

navegación, caótica. El AUV sube y baja por la flotabilidad variable, pero no ‘vuela’. Es ingobernable.

El simulador nos permite probar distintas configuraciones de velocidad y sincronismo en las maniobras de control, y ver su eficacia.

15.2.3.2 Valores de consigna para determinar un vuelo uniforme.

Los controles que tenemos en nuestro AUV Glider, nos permiten establecer con precisión 2 cosas:

1 Lastre líquido incluido, y

2 posición de la masa móvil

Eso genera fuerzas y momentos que crean el movimiento del Glider. Cuando el Glider está planeando en modo estacionario, el vuelo se caracteriza por valores constantes de:

- Angulo de planeo
- Velocidad

Ambos pueden elegirse con independencia uno de otro. Pero no todas las sendas de planeo son posibles. Depende de la hidrodinámica del Glider.
La posición de la masa móvil afecta exclusivamente al ángulo de asiento, pero como el lastre líquido variable está en proa, un cambio de flotabilidad también altera el asiento. Si alguna de las entradas (lastre y posición de masa móvil) produce variación en más de una salida (ángulo de planeo y Velocidad), decimos que hay *acoplamiento* entre entradas y salidas. Ese es nuestro caso. Pero basta con compensar la interacción cruzada o acoplada entre los controles y las salidas, y podremos modificar con independencia ángulo de planeo o velocidad, sin que se altere el otro.

15.2.3.3 Características de planeo de nuestro GLIDER, y comparación con otro conocido (SLOCUM)

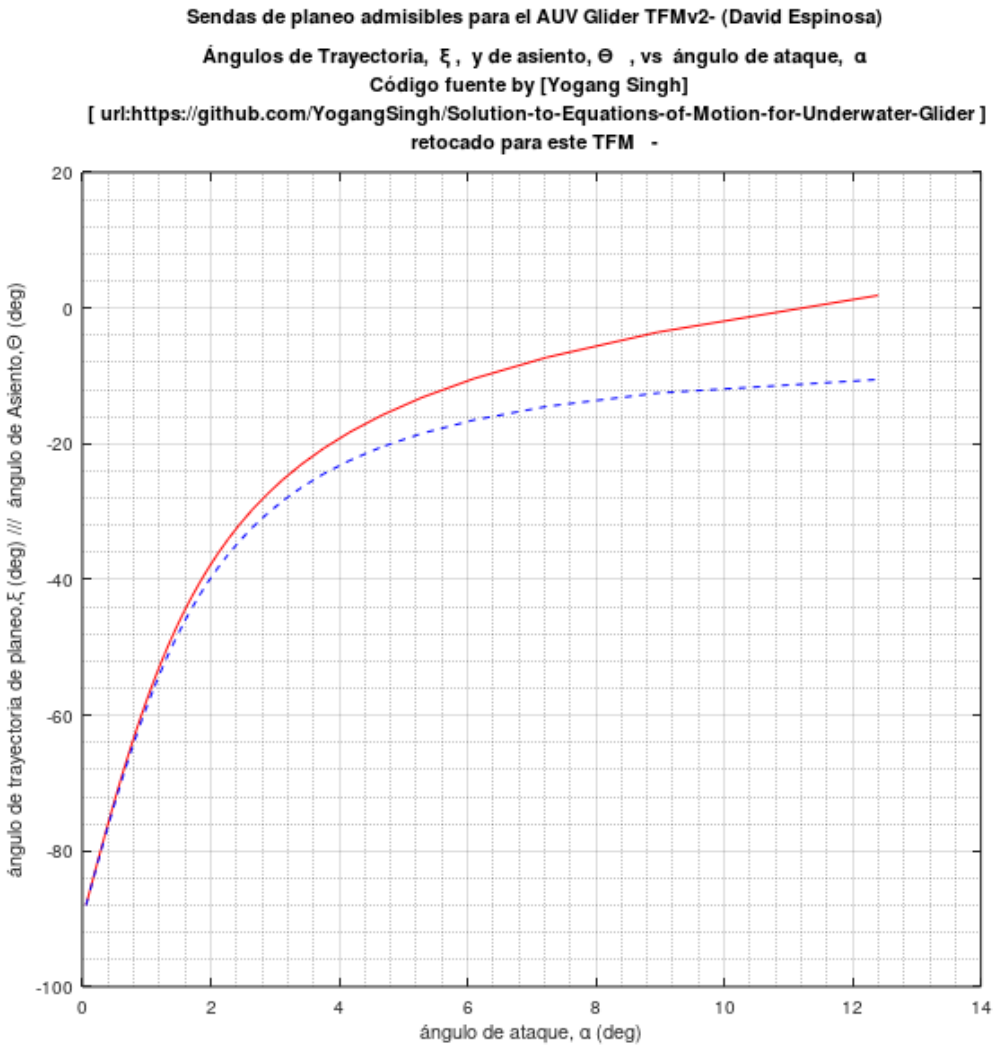


Ilustración 28 Sendas de planeo posible y ángulo de planeo. Fuente Propia.

Esta figura, de elaboración propia, es la que corresponde a NUESTRO Glider, y muestra las sendas de planeo posibles y los ángulos de ataque correspondientes. También reproducimos la gráfica del SLOCUM, sacada de la referencia **[Graver 2005]** .

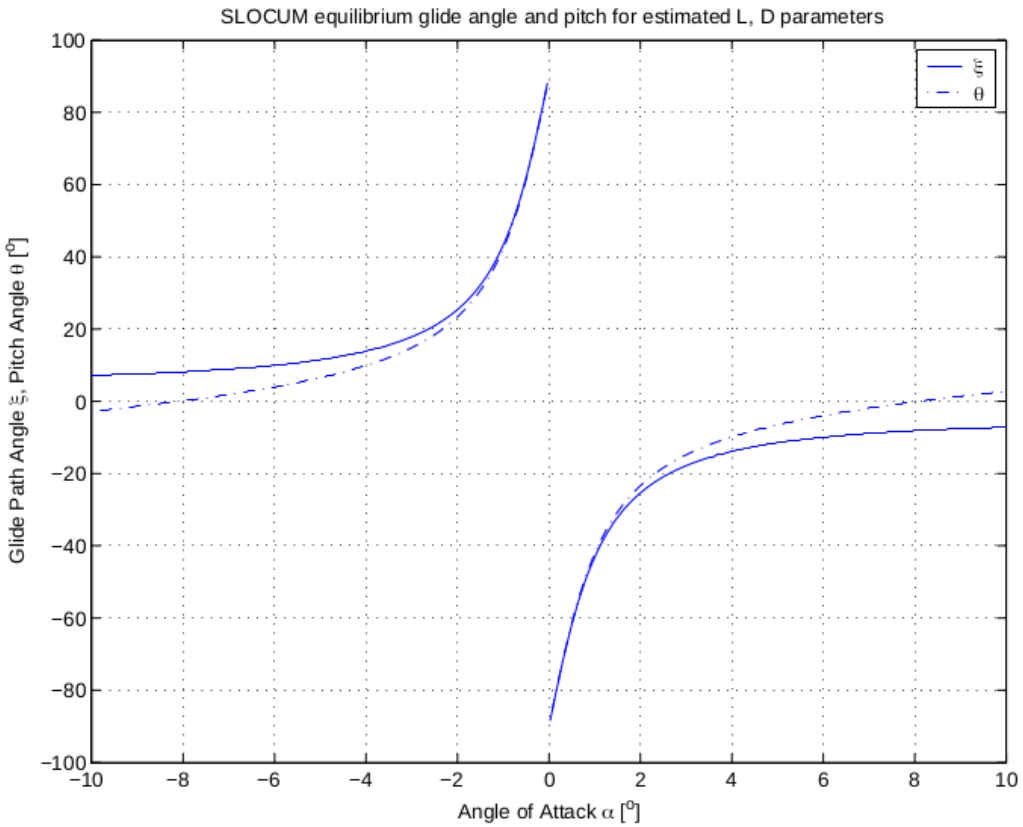


Figure 5.7: Equilibrium glides using lift and drag estimated from reference data.

Ilustración 29 SLOCUM. Graver 2005

Comparando las dos figuras puede concluirse que nuestro Glider planea un poco peor que el SLOCUM. Su ángulo de planeo (respecto a la horizontal) es unos 5º mayor que el del SLOCUM cuando el ángulo de ataque en ambos es de 6º.

15.3 Estrategia de la navegación. Vigilancia y control.

15.3.1 La maniobra de Inversión.

La maniobra más crítica durante la navegación es la de inversión de movimiento. Cuando se pasa de movimiento descendente a ascendente, o viceversa. En esos casos es crucial que los movimientos de los elementos de control sean ejecutados a su tiempo y a la velocidad debida, lo cual se debe estudiar con simulaciones y ensayos reales combinados.

15.3.2 El resto de la navegación

Una vez terminada una maniobra de inversión del movimiento, no hay que hacer nada, salvo monitorizar algunos sensores de la navegación:

- Profundidad,
- Velocidad de cambio de profundidad
- Angulo de asiento
- Velocidad estimada

y esperar a que llegue (es previsible) el momento de la maniobra siguiente.

Por tanto, la mayor parte del tiempo de navegación es monitorizar el ‘estado’ vigilando los parámetros de navegación y del funcionamiento de la nave. 7

En la vigilancia puede ocurrir que surja la detección de alguna anomalía. Es en ese caso cuando hay que actuar según los protocolos previstos. Debe tenerse bien estudiados una colección de casos y sus respuestas, y el sistema ha de ser capaz de valorar las prioridades cuando se den más de una anomalía simultáneamente.

La gestión de las incidencias. Programación dirigida por eventos.

15.3.3 Programación dirigida por eventos.

Hay una buena parte de incertidumbre respecto a ‘cuando’ y ‘cuales’ sucesos van a ocurrir. Por ello, la labor de vigilancia puede ser agotadora o consumir muchos recursos si no se trata adecuadamente.

En un proceso automático, o asistido por procesador, es un programa el que hace el seguimiento y las acciones debidas.

El paradigma de programación tradicional, de tipo imperativo y secuencial o por procedimientos, no es adecuado para crear un código eficaz en este tipo de operativa. Aquí el método más apropiado es el de la programación Orientada por Eventos.

Existe bibliografía que avala esta forma de control de la navegación de los Gliders. En el ANEXO se trata esto con más detalle.

15.4 Máquinas de estados finitos

Una forma de implementación sistemática de la programación dirigida por eventos es la creación de máquinas de estados finitos que se comunican entre sí, y cada una busca un objetivo, siguiendo unas reglas, y teniendo en cuenta el estado de las otras máquinas. El resultado es como un sistema multitarea, en el que todo ocurre de modo simultaneo, pero sin colisiones ni atascos. Ninguna tarea 'cuelga' el sistema ni impide la evolución de las demás tareas. Y no se requiere que el procesador sea multinúcleo o multitarea. Esa eficacia y óptima utilización de los recursos del procesador sería imposible de lograr con un estilo de programación 'tradicional'.
En el entono Arduino existen librerías que facilitan esta forma de control de procesos. También para otros procesadores (como el ESP32) existen herramientas similares. Hemos encontrado algunos ejemplos:

- **[Automaton]**, librería para el entorno ArduinoIDE, en código C++. Tambien vale para el procesador ESP32 y placas basadas en él como NodeMCU-ESP32s.
- **[ESP_EVENT]**, librería creada por el fabricante del chip ESP32, para su uso con el entorno IDE propio del ESP32

A medida que avanzamos en nuestro estudio, vamos descubriendo que existen algunos productos muy orientados al Internet de las Cosas (IoT, Internet of Things) y que utilizan con frecuencia la programación dirigida por eventos. El procesador ESP32 es un producto bién valorado en este ámbito.

15.5 Un ejemplo de aplicación a un AUV GLIDER.

La figura adjunta, extraída de la referencia **[A High Accuracy Navigation System for a Tailless Underwater Glider]** es el grafo que representa la **máquina de estados** simple que gobierna el sistema Glider descrito en esa referencia.

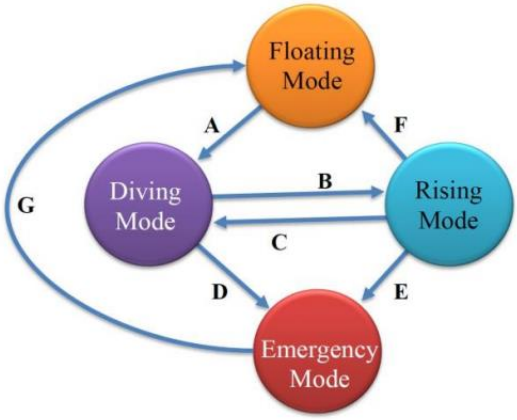


Fig. 3. Functional Modes.

Ilustración 30 Representación de estado simple de gobierno del GLIDER. Tailless Underwater Glider.

(pie de foto) La **máquina de estados finitos** (FSM, Finite State Machine) representada en la figura es un modelo del sistema de navegación de un Glider.

En el ANEXO se explica con más detalle cómo operan estos algoritmos.

15.6 Revisión del entorno de electrónica Arduino frente a otros más adecuados: NodeMCU-ESP32s.

A media que vamos estudiando y conociendo las particularidades de la gestión de la navegación de un AUV Glider, nos vamos dando cuenta de que la plataforma Arduino, muy adecuada para iniciarse en la electrónica, robótica, etc., debido a la gran comunidad existente a su alrededor, es un poco obsoleta para los retos que se plantean en nuestro AUV Glider.

Eso nos ha conducido a un cambio de dirección en el estudio. La placa de desarrollo hacia la que hemos cambiado el rumbo es la NodeMCU-ESP32s. salió al mercado en 2018, y tiene grandes cualidades:

- bajo precio (unos 13€, la mitad que un Arduino MEGA 2560)
- pequeño tamaño (¼ del tamaño del Arduino MEGA 2560)
- 2 procesadores a 32 bits, capacidad multiproceso-multitarea
- FLASH RAM (no volátil) de 4 MB. (ESP32-WROOM-32 (ESP-WROOM-32) integrates 4 MB of external SPI flash.)
- Bt y WiFi integrados
- programable en Python, C++ (como Arduino), LUA, etc.…

Ese cambio nos ha permitido avanzar en otros aspectos.

Las comunicaciones en superficie entre el AUV Glider y el Usuario se realizan vía WiFi, con un Smartphone normal en el lado del Usuario.

La placa NodeMCU-ESP32s establece un Punto de Acceso (AP, AccessPoint) y sirve una página Web con la Interfaz de control GUI.

El usuario se conecta al AP y maneja la GUI en su navegador habitual. En nuestro caso hemos trabajado con Firefox - Motzilla.

Desde la GUI se puede programar una misión para el AUV Glider, y este la ejecutará de modo autónomo, basado en el plan de navegación y sus sensores de rumbo, profundidad, etc.

La radiación Wifi, de 2.4 GHz, se atenúa muy rápido al penetrar en el agua, de modo que las comunicaciones solo son posibles en superficie, mientras se prepara la misión.

También el GUI permite el telemando de los motores y servos, y muestra en pantalla dato tales como Tensión de baterías, intensidad consumida por el motor, rumbo, escora, trimado, temperatura en el interior del AUV, etc.

Todo este nuevo mundo es accesible desde nuestro teléfono móvil
Desde la pantalla del móvil se pueden mover los motores simplemente arrastrando el dedo por las barras deslizantes preparadas para ello., y los sensores actualizan los datos en pantalla.

Las claves que han desencadenado el cambio de CPU a lo largo de este estudio son las facilidades de desarrollo en ESP32 de los siguientes aspectos:

- capacidad de Programación Dirigida por Eventos, librería ESP_EVENT para ESP32, desarrollada por el fabricante de ESP32: EXPRESSIF
- WIFI integrado en la placa
- capacidad de crear un AP en modo server de páginas html, al que se puedan conectar diversos clientes, sin necesidad de enlace a Internet. (Facilidad de los tutoriales para implementar estas funciones)
- Precio y tamaño muy contenidos.
- Compatibilidad de programación en C++ con el Arduino IDE
- Compatibilidad con muchas de las librerías de Arduino, y con los dispositivos que se usan con el entorno Arduino (sensores, motores, etc...).

16. Collage de Fotos del AUV Glider construido

Al límite del plazo de entrega del TFM, tenemos el AUV Glider en el inicio de las PRUEBAS del producto terminado, y confiamos en seguir mejorándolo, pero ya no queremos demorar más esta presentación.

En estas imágenes se muestran algunos aspectos del producto que hemos construido como parte del estudio y con la finalidad de seguir estudiando este tipo de naves.

Ilustración 31 Planeador submarino AUV GLIDER montado1. Fuente propia.

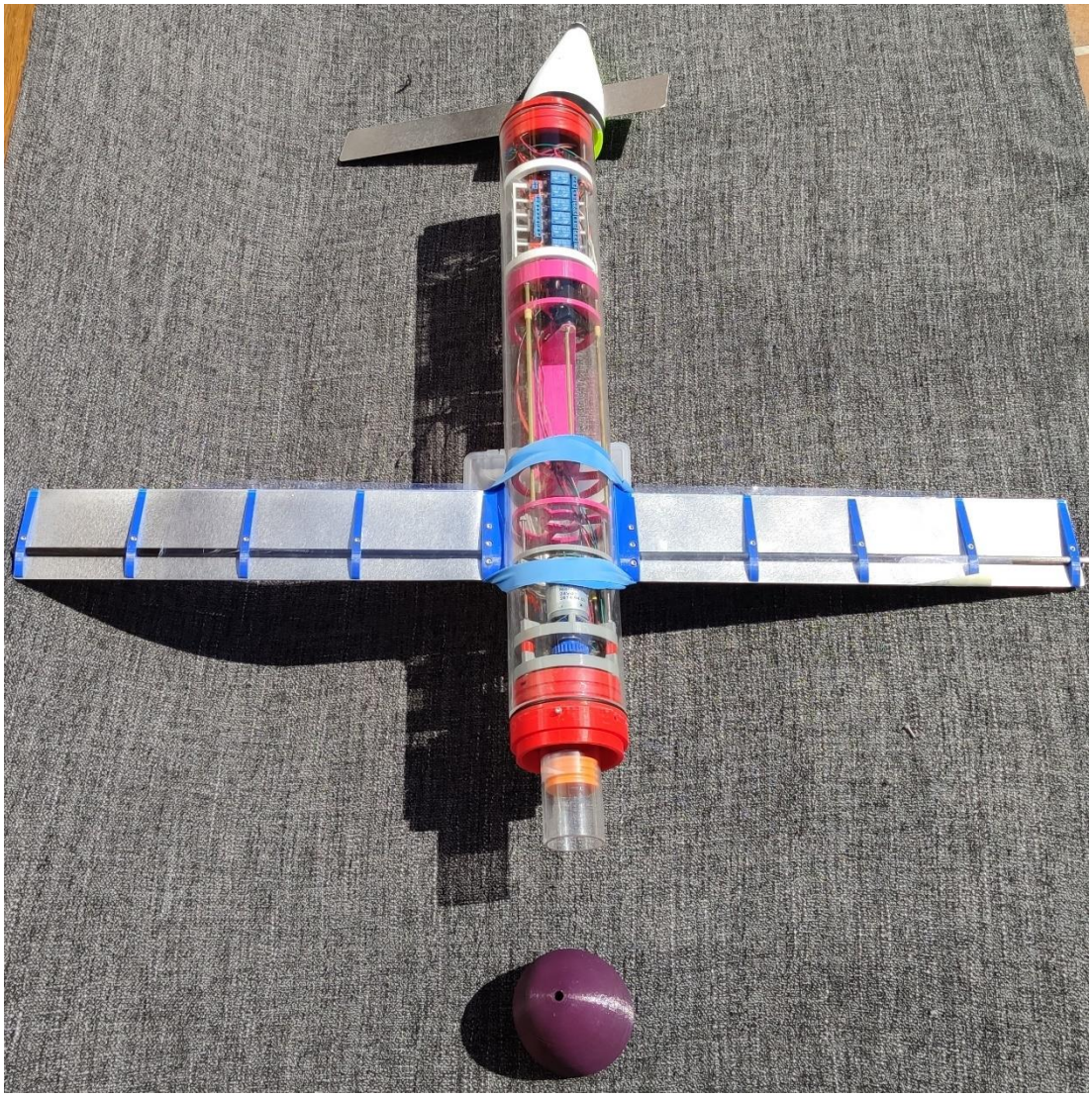


Ilustración 32 Planeador submarino AUV GLIDER montado 2. Fuente propia.

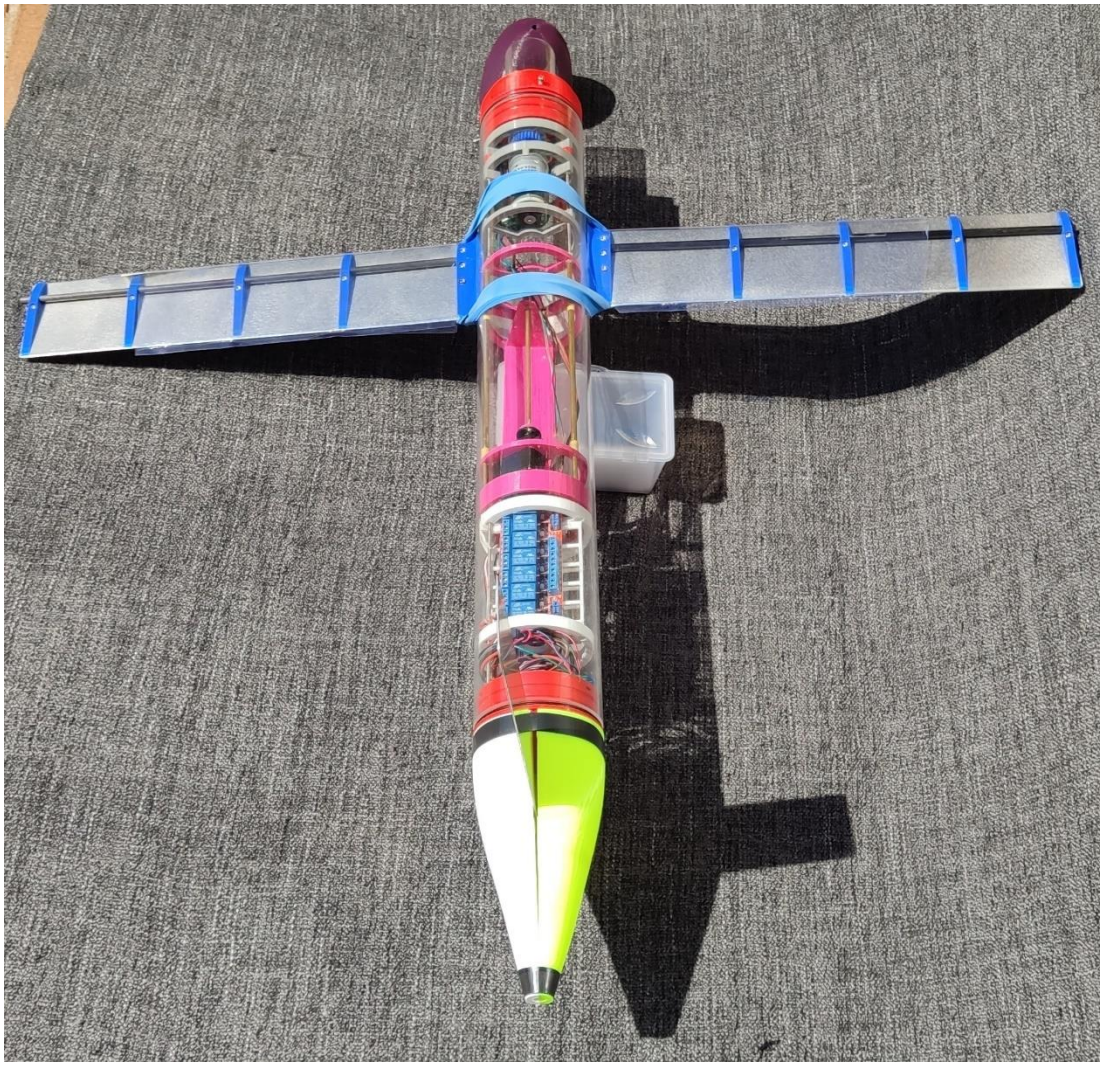


Ilustración 33 Planeador submarino AUV GLIDER montado 3. Fuente propia.

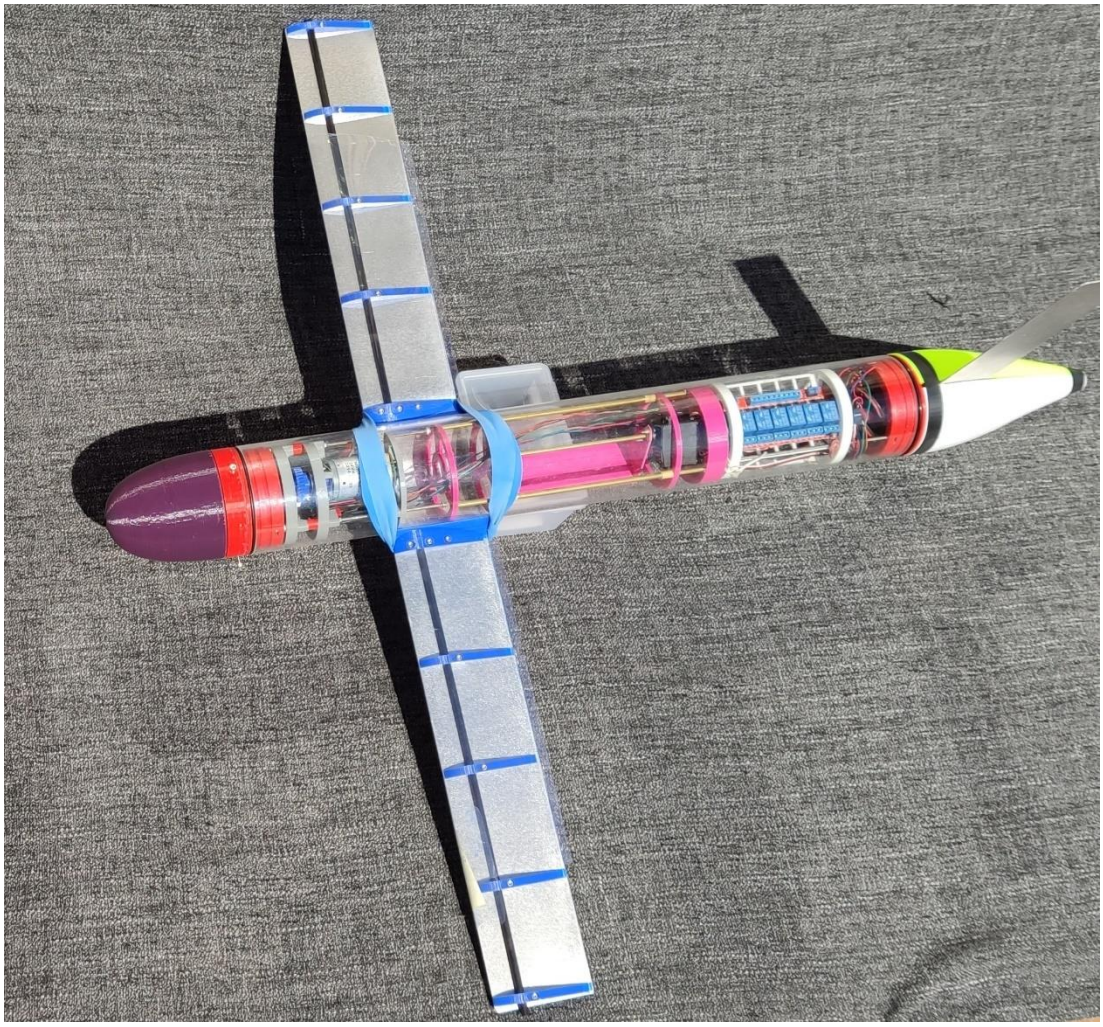


Ilustración 34 Planeador submarino AUV GLIDER montado 4. Fuente propia.

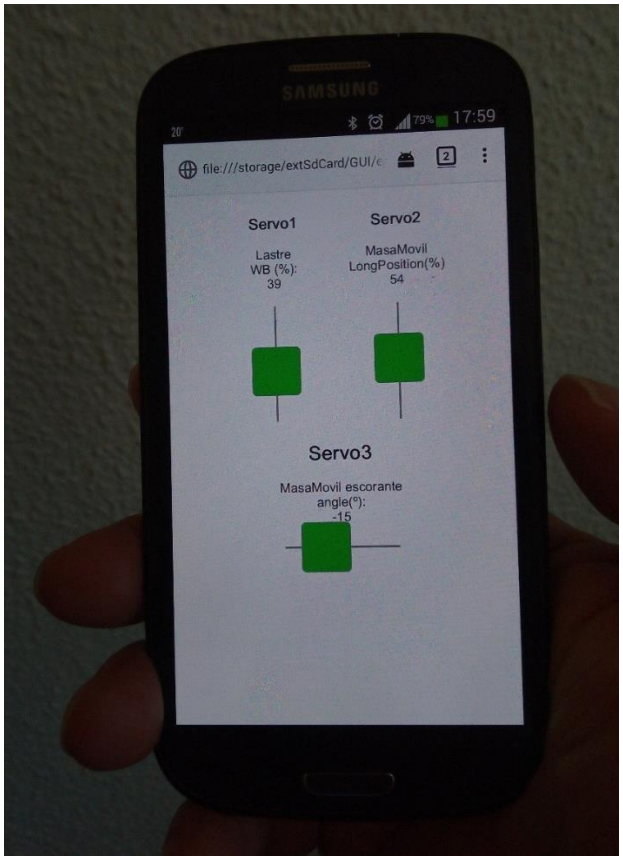


Ilustración 35 Pruebas del GUI (Graphical User Interface) desde un movil. Fuente propia.

Ilustración 36 Submarino AUV perspectiva 1. Fuente propia

Ilustración 37 Submarino AUV perspectiva 2. Fuente propia

Ilustración 38 Submarino AUV perspectiva 3. Fuente propia

Conclusión.

Este trabajo se inició como un estudio con la intención de llegar a construir un AUV Glider operativo con nuestros propios medios. No podía ser un proyecto, porque no teníamos referencias para proyectar. Desconocíamos todo sobre los AUV Glider. Por eso la única posibilidad de avanzar en el conocimiento de estos productos era explorar en varias direcciones e ir siguiendo por intuición los caminos que parecían más convenientes, prometedores o accesibles.
Eso nos ha conducido en la práctica a alcanzar un producto muy estudiado, bien diseñado, construido y funcional, que incorpora todos los ingredientes que podíamos imaginar.

Para desarrollarlo se ha fijado un objetivo, el cual pretendía que el producto fuera funcional para poderlo realizar de forma física. Parte de esto se alcanzó en un estudio TFG previo, donde se desarrolló el diseño y la construcción de la parte física. Una vez obtenida la construcción de la parte física, partimos con unas bases para el desarrollo de la parte electrónica y de control.

Finalmente hemos construido, en paralelo al estudio, un AUV Glider real y estamos en fase de pruebas del producto acabado

El trabajo ha sido triple:

- Estudio – investigación – aprendizaje
- Proyecto - aunque no formalizado, ni en formato convencional. Debido a que estaba sujeto a demasiados cambios, y además el destinatario-constructor somos nosotros mismos, autores de este trabajo, podemos permitirnos esa informalidad, en aras de una mayor flexibilidad, y en su lugar nos aprovechamos de nuestro progresivamente creciente conocimiento del producto en estudio
- Construcción.

Se ha conseguido que, por medio de la electrónica y con unas rutinas de navegación, el GLIDER es capaz de recorrer unas distancias determinadas. En caso de tener un problema mientras no se pueda tener comunicación con él, se ha realizado una rutina de emergencia para que el dispositivo sea capaz de volver a la superficie y mandar una señal de ubicación.

Para todas las rutinas, se le dotó de un CPU y unos relés capaces de desconectar equipos (esenciales o no), teniendo una batería principal y una de apoyo o secundaria en caso de pérdida de energía.

Basándonos en otros estudios hemos podido ayudarnos a crear y verificar la teoría de vuelo con los parámetros de nuestro submarino, comprobando que la teoría de vuelo resultante es correcta y los datos arrojados por el sistema de cálculo son coherentes.

Las circunstancias que nos han permitido alcanzar este resultado han sido especiales y atípicas. Posiblemente irrepetibles, pero ya está hecho.
La satisfacción y el logro han merecido la pena.

Nos hemos quedado a las puertas de las PRUEBAS FINALES del producto terminado, y confiamos en seguir mejorándolo.

Bibliografía

Páginas webs.

1- Martyn Cyrrey and powerd by WordPres , 06-agosto-2020 http://www.martyncurrey.com

2- Plataforma arduino. Aprendiendo arduino wordpress. 06-agosto-2020. https://aprendiendoarduino.wordpress.com

3- Plataforma de arduino. Adafruit. 6-agosto2020. https://learn.adafruit.com

4- Plataforma Wikipedia. 6-agosto 2020. https://es.wikipedia.

5- Plataforma shieldlist, 06-agosto 2020.http://shieldlist.org/

6- Plataforma de contenido. 20-08-2020 http://srferrete.es/

7- [Singh 2018] . Un código MATLAB, de libre accceso, que implementa el modelo matemático descrito en [Grave, 2005]. 04/09/2020 https://github.com/YogangSingh/Solution-to-Equations-of-Motion-for-Underwater-Glider/

Documentos.

8- [Graver, 2005] [Underwater Glider dynamics and control. 2005, J. G. Graver] Tesis doctoral. "Se descubre que esta fuente es la más completa y detallada en el desarrollo matemático del modelo, pues es la fuente de muchas otras."

9- **[ODE – simulation with MATLAB, OCTAVE and SCILAB]** Simulation of ODE/PDE Models with MATLAB#, OCTAVE and SCILAB. Philippe Saucez, Alain Vande Wouwer, Carlos Vilas, Ed Springer, Scientific and Engineering Applications

10-**[A High Accuracy Navigation System for a Tailless Underwater Glider]** . Enrico Petritoli, Fabio Leccese . IMEKO International Conference on Metrology for The Sea Naples, Italy, October 11-13, 2017 . una descripción de navegación controlada por eventos.

11-**[Docking for an Autonomous Ocean Sampling Network]. IEEE JOURNAL OF OCEANIC ENGINEERING, VOL. 26, NO. 4, OCTOBER 2001** descripción de navegación controlada por eventos

12-Safety of Tourist Submersibles. Committee on Assessing Passenger Submersible Safety Marine Board. Commission on Engineering and Technical Systems. **National Research Council. NATIONAL ACADEMY PRESS.** Washington, D.C. 1990 https://www.nap.edu/catalog/1744/safety-of-tourist-submersibles [PDF] https://www.nap.edu/download/1744 último acceso al sitio web en 5-7-2020.

13- **[SECTION 18 Autonomous Underwater Vehicles (AUV) (2014)]** ***ABS RULES*** FOR BUILDING AND CLASSING **UNDERWATER VEHICLES**, SYSTEMS AND HYPERBARIC. FACILITIES . 1 ene. 2019 ww2.eagle.org › 7-underwatervehicles › uwv-jan-2019 (PDF) último acceso al sitio web en 5-7-2020

Anexo

Modelo matemático del AUV GLIDER para las simulaciones

Simulación y control del AUV GLIDER

Tabla de contenidos

Contenido

1 Simulaciones y trabajos relacionados con sistemas dinámicos y su control. Software especializado.

Entre los programas existentes para estudiar sistemas dinámicos controlables, uno de los más conocidos y usados actualmente es MATLAB.

MATLAB es muy popular, pero es privativo (su uso requiere el pago de una licencia). Dispone de muchos seguidores y hay mucha bibliografía y ejemplos para resolver gran cantidad de cosas.

Por contra, existen otros programas, como OCTAVE and SCILAB, tienen menos seguidores y menos fama, pero que son OpenSource (y gratuitos). Estos programas Open Source, pretenden 'emular' a los privativos líderes del sector. Suelen buscar la compatibilidad y la similitud con los privativos en todas las funcionalidades y menús, y así poder captar seguidores más fácilmente. Nosotros hemos acordado usar estos programadas Open Source, y podemos decir que su compatibilidad, modo de uso y capacidades tienen poco que envidiar del MATLAB. No obstante, existen algunas diferencias, que en la práctica hemos salvado sin problema con la ayuda de algunas referencias adecuadas, como **[ODE – simulation with MATLAB, OCTAVE and SCILAB]**

Esta referencia nos ha aportado información sobre las similitudes y diferencias entre los entornos de simulación más habituales, y muestra ejemplos de los mismos problemas resueltos con cada uno de esos programas, señalando las conversiones o modificaciones necesarias para allanar tales diferencias.

En este punto, y antes de seguir, conviene aclarar una posible pregunta ¿qué es la **simulación**? En nuestro caso se trata de resolver numéricamente unas Ecuaciones Diferenciales que representan matemáticamente con suficiente perfección la Mecánica – Física que rige la navegación del Glider.

Simplemente:

- tener unas ecuaciones diferenciales adecuadas para nuestro problema (las leyes del movimiento de planeo según la geometría del Glider).
- resolverlas numéricamente para diversos valores de los parámetros representativos de la geometría y el movimiento.

Así, por comparación entre unos y otros casos, podremos saber cuáles son los efectos de modificar cualquiera de esos parámetros, y encontrar cuales son los parámetros más convenientes para el diseño del Glider.

La simulación es una grán herramienta para el diseño. Permite abreviar el proceso de perfeccionamiento del diseño por prueba-error-corrección.

En lugar de construir físicamente una versión del Glider y hacerlo navegar midiendo sus prestaciones, para posteriormente decidir modificaciones y volver a construir y probar la versión modificada (y reiterar el ciclo), la simulación permite hacer esos cambios 'virtualmente' (en las ecuaciones) y obtener los nuevos resultados, 'solo' con cálculo automático de ordenador.

La primera parte del problema, tener unas ecuaciones diferenciales suficientemente representativas del movimiento del Glider, es algo que resolveremos más adelante.

Sobre la segunda parte del problema, solución numérica computacional de las EDO (Ecuaciones Diferenciales Ordinarias, ODE en inglés), confiamos en tener una buena ayuda gracias a los programas Open Source citados. En la *tabla* siguiente, de elaboración propia pero inspirada en la referencia **[ODE – simulación …..]**, se remarca el paralelismo entre esas opciones: privativa y OpenSource. En la tabla se han puesto a la misma altura (misma fila) los programas que son suficientemente 'equivalentes' (en apariencia, funcionalidad y modo de uso).

privativo		OPENSOURCE		
MATLAB	Simulink	OCTAVE	SCILAB	Xcos

Existe grán similitud (equivalencia) entre MATLAB y OCTAVE, de tal modo que los archivos .m son compatibles en ambos.

Sin embargo, entre SCILAB y MATLAB el paralelismo no es total. La funcionalidad de MATLAB puede reproducirse en SCILAB, pero requiere algunas conversiones y cierto conocimiento de ambas aplicaciones para hacerlo. No obstante, existe una funcionalidad automática en el menú de SCILAB para 'convertir' archivos .m al formato de SCILAB.

Como vemos, *Simulink* tiene un equivalente que es *Xcos*. Y además, del mismo modo que *Simulink* es un *Addon* (un complemento *añadido*) de **MATLAB**, *Xcos* lo es de **SCILAB**. Esa dependencia también se ha querido señalar en la tabla.

Podemos decir que la funcionalidad del conjunto (privativo) ***MATLAB+Simulink*** puede ser lograda con el conjunto (open Source) ***OCTAVE+SCILAB+Xcos***

La conclusión es que existen alternativas en OpenSource para las dos aplicaciones MATLAB y Simulink líderes en el ámbito del Software para Estudio de Sistemas dinámicos. En este trabajo hemos usado tales alternativas con éxito, pues queremos apostar por el OPEN SOURCE (y además, en cualquier caso, no disponemos de licencia MATLAB personal o particular).

2 El AUV Glider como Sistema Dinámico.

Tanto los buques de superficie como los sumergibles tienen un comportamiento ante los balances que se puede describir con una ecuación diferencial de segundo orden de la escora en función del tiempo, semejante a la del péndulo compuesto o físico.

Ese modelo matemático en forma de EDO representa un Sistema Dinámico que conocemos bién. y sabemos que es estable solo si un determinado parámetro lineal es positivo y no demasiado pequeño. El parámetro de referencia para la estabilidad, asimilable a la longitud del péndulo físico, es la altura metacéntrica, GM, (en buques de superficie), o la distancia GB (en sumergibles)

En todo caso G (centro de gravedad) debe estar por debajo de los otros puntos: M (metacentro), o B (centro de carena o flotabilidad – Boyancy en inglés).

En esos casos, y asumiendo que no hay fuerzas excitadoras exteriores, el sistema (vehículo balanceándose) es estable (tiende a una posición de equilibrio), y las oscilaciones previas a llegar a esa posición estable son amortiguadas. Tanto el periodo de balance como el grado de amortiguamiento de ese movimiento quedan descritos por 3 parámetros característicos relacionadas con las fuerzas de restauración, las fuerzas disipativas de energía, y las fuerzas inerciales de masa.

Con esas premisas, el sistema buque es estable ‘por diseño’ siempre que G (Centro de Gravedad, cdg,) esté por debajo del otro punto característico: Metacentro M (en buques de superficie), o centro de Carena B (en los sumergibles).

Centrándonos en el sistema sumergible que estamos estudiando, el Glider, si el G (Centro de Gravedad, cdg) está por debajo de B (centro de Carena) el sistema

es estable, y las fuerzas debidas a la viscosidad del fluido agua, y a las alas, hacen que el movimiento se amortigüe con rapidez.

Antes de seguir avanzando, vamos a aclarar que en los sumergibles no hay diferencia entre estabilidad longitudinal o estabilidad transversal. En ambos casos la relación entre el momento inclinante y el ángulo resultante de inclinación depende de un mismo parámetro GB.

Momento vertical de fuerzas = peso real * GB* seno (ángulo de inclinación)

y ocurrirá lo mismo con cualquier otro plano vertical que se use como referencia para analizar la relación entre momento aplicado e inclinación resultante.

En lo sucesivo tendremos en mente que estamos estudiando la estabilidad longitudinal, conforme al modelo 2D de movimiento que hemos elegido para el estudio

Por tanto, en la práctica, en estos sistemas toda la teoría de control se reduce a asegurar valores 'adecuados', según enseña la Teoría del Buque - estabilidad.

No obstante, lo anterior, en sistemas novedosos, menos convencionales, donde pueden existir fuerzas adicionales de propulsión o de gobierno que generen acciones de gran magnitud, puede alterarse esa estabilidad, y es entonces cuando puede cobrar interés aplicar la teoría de control y de estabilidad genérica de sistemas.

Si los actuadores de nuestro GLIDER generan cambios grandes y repentinos en el sistema de fuerzas (flotabilidad y momento escorante) que sirven para su control, pueden crear una dinámica no deseable. Ahí entra en juego la elección de la velocidad de los actuadores, y la magnitud de los momentos aplicados.

Afortunadamente en nuestro caso, los GLIDERS, son estables en un rango relativamente amplio de valores del ángulo de ataque alfa de las alas respecto al fluido en el que se mueve. Ese ángulo es el que caracteriza una determinada posición de vuelo.

Todo esto nos lleva a una conclusión final, que por razones de cultura naval conocemos ya. El submarino será estable, por tener el punto G por debajo de B. Y si la construcción resultase de otro modo, se deberán adaptar las masas y lastres para conseguir que sea así.

En terminología de la teoría de control y sistemas diremos que el sistema es estable en lazo abierto.

Además, la magnitud de los amortiguamientos y el brazo de estabilidad hacen que el sistema tenga una relación de reducción de amortiguamiento del orden de un 50 %, o mayor, en cada oscilación.

Todo esto lo conocemos a priori por cultura naval. Sin embargo, nos hemos acercado a este estudio desde una visión más amplia, asumiendo que podríamos estar ante un sistema menos conocido, donde la estabilidad pudiera ser precaria, y su control un problema.

Adelantamos también que el control, en nuestro caso, se reduce a mover a la velocidad adecuada (ni poca ni mucha) las masas de control, que serán 2: lastre (variable) y masa móvil

3 Modelo matemático del GLIDER para las simulaciones

Pero nuestras decisiones suelen estar condicionadas. Mientras el Glider no esté disponible para pruebas, no tenemos la opción de identificar el sistema real. Por tanto, para ganar tiempo en paralelo con la construcción, debemos obtener un modelo matemático teórico sobre el que iremos estudiando el comportamiento del Glider en navegación.

En la bibliografía existen estudios donde se han obtenido modelos adecuados. Veamos cómo se ajusta nuestro Glider a alguno de esos modelos.

3.1 Descripción de la forma de navegar del GLIDER, orientada a establecer el modelo.

El Glider de referencia que estamos estudiando es un cuerpo fusiforme dotado de un ala fija en su zona media. Además, dispondrá de un plano vertical en la cola, para estabilizar el rumbo. El Glider y los planos alares no tiene partes móviles externas. El volumen desplazado es constante, y la posición del centro de ese volumen es invariante en los ejes de referencia fijados al GLIDER. En la siguiente figura se muestra (sin alas) una visión esquemática del modelo

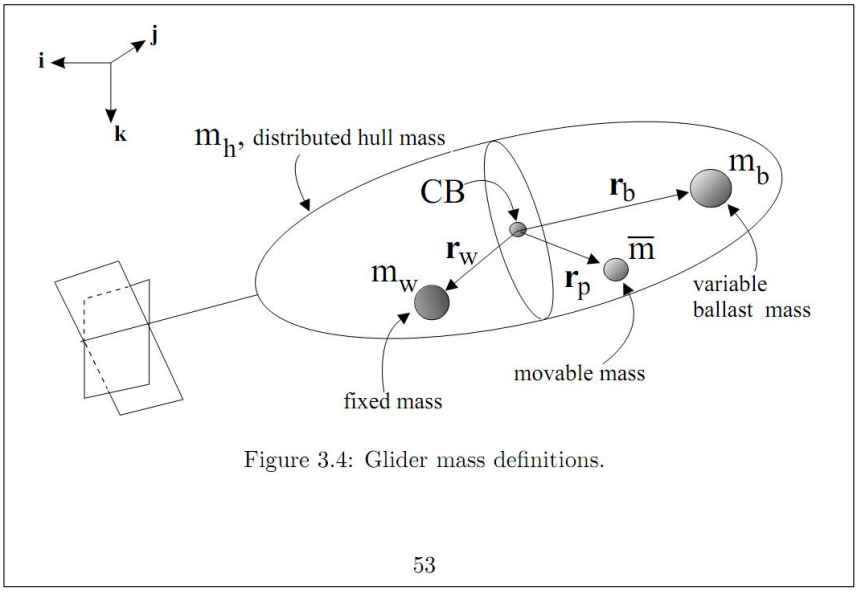


Figura sacada de **[Graver , 2005]** , pag 53

El control de la navegación se basa en dos actuadores interiores que cambian el estado de masas del Glider, alterando dos masas asumidas puntuales:

- una masa, $\mathbf{m_b}$, de magnitud variable, pero de posición fija: el lastre (**b**allast en inglés). Comúnmente es líquido que se trasiega entre el interior y el exterior del Glider, y que modifica la flotabilidad del conjunto, dando una resultante vertical, hacia arriba o hacia abajo, que es la causa primera del movimiento del Glider. Esa es la forma de modificar la flotabilidad por el principio de Arquímedes. El mismo principio que también utiliza la naturaleza con la vejiga natatoria de los peces.
- una masa móvil interior, $\overline{m}$,capaz de moverse longitudinal y transversalmente, y que con ello modifica la posición del Centro de Gravedad (CDG) del conjunto del Glider.

El movimiento relativo del Glider en el líquido que le rodea se caracteriza por un vector velocidad. Ese movimiento dá lugar a fuerzas hidrodinámicas, que dependen de la geometría del Glider, y de ese vector velocidad.

El actuador de la masa móvil puede cambiar su posición, y con ello el CDG. Por ello, y para mantener la condición de equilibrio, también cambia la pose o '**actitud'**, la posición presentada por el Glider ante el vector velocidad. En consecuencia, cambian las fuerzas hidrodinámicas y también el movimiento.

El proceso cumple las leyes de la física y la mecánica, de modo que es posible representarlo por las ecuaciones diferenciales clásicas que relacionan fuerzas y aceleraciones. Seguidamente vamos a presentar varios modelos de distinto nivel de completitud y complejidad.

3.2 Modelo1, (3D). Movimiento del Glider en el espacio tridimensional.

En la referencia **[Graver , 2005]** se plantea un modelo completo del movimiento en las 3 direcciones del espacio. Se describe en el cap 3 de esa referencia, y el resultado es un sistema de ecuaciones diferenciales con matrices muy grandes 15x15 que describen un espacio de estados con un vector de estados de 15 componentes, y un vector de entradas de 4 componentes.

Las entradas son:

- 3 componentes cartesianas de la fuerza, relativa al Glider, que actúa sosteniendo la masa móvil en su posición (3 coordenadas en ejes del cuerpo)
- el ritmo de bombeo de la masa de lastre líquido. (kg /s)

Las masas involucradas en el modelo incluyen también la masa del líquido exterior que acompaña al cuerpo en su movimiento. Esa ***masa añadida*** de fluido tiene fundamento real, y queda reflejada en el modelo matemático. (ver pag 63 de la referencia **[Graver , 2005]**).

El conjunto de ecuaciones, variables y parámetros está muy bien definido en la referencia, pero su extensión es grande. Maneja 15 estados, lo que supone trabajar con matrices de orden elevado (productos e inversiones de matrices con necesidad de grán potencia de cómputo). El propio autor lo usa solo como base para posteriores simplificaciones hacia modelos más prácticos. Nosotros solo lo mencionamos.

3.3 Modelo2, (2D) Movimiento del glider en un espacio bidimensional: en plano vertical y masas centradas

Siguiendo con la referencia **[Graver , 2005]**, el autor procede a simplificar algo ese modelo. La primera aproximación será restringir la navegación a un plano vertical, coincidente con el plano de crujía del Glider, asumido simétrico y adrizado. Eso reduce el movimiento a dos dimensiones lineales, y un giro en ese plano. Se reduce el número de componentes vectoriales a tratar, pero persisten las no linealidades, y las demás complejidades. El vector de estados en este sistema tiene 11 componentes.

En el capítulo 4 de esa referencia **[Graver …]** hacen esas simplificaciones. Y también añaden 2 simplificaciones más: la masa variable, $\mathbf{m_b}$, la sitúan en el centro del volumen del Glider, **C**, y además eliminan la asimetría de masas inicialmente considerada en el modelo 3D, y materializada con la masa, $\mathbf{m_w}$, que ahora se hace nula.

No obstante, las conclusiones que se deducen de trabajar en el modelo simplificado son representativas para cualquier Glider, y sobre el modelo final al que se llega se puede volver a restituir, si se desea, la posición del lastre.

Ahora las entradas de control del sistema serían 3:

- 2 componentes cartesianas de la fuerza de reacción contra el Glider, que actúa sosteniendo la masa móvil en su posición (2 coordenadas en ejes del cuerpo)
- el ritmo de bombeo de la masa de lastre liquido

y el vector de estados tiene 11 componentes.

Este sistema queda completamente descrito para nuestro caso, pero tiene todavía algunas dificultades prácticas de manipulación matemática para ser resuelto. Esas dificultades son salvables con un software adecuado tipo ***MATLAB***, (o ***SCILAB*** u ***OCTAVE***, que son OpenSource y capaces de hacer lo mismo) y con un ordenador de suficiente potencia.

Con ello podríamos hacer algunas simulaciones y ver cómo responde el Glider a las acciones. Podríamos encontrar algunas trayectorias estables, y comprobar que determinados …

1 - pares de control: (posición-masamovilprimaria ($\mathbf{r_p}$) , masalastre ($\mathbf{m_b}$,)), dan lugar a tales

2 - trayectorias de planeo: (ángulo de planeo (seta, $\boldsymbol{\xi}$), velocidad ($\mathbf{v}$)), o cuales

3 - condiciones de navegación: (ángulo de trimado (teta, $\boldsymbol{\theta}$), ángulo de ataque (alfa, $\boldsymbol{\alpha}$)),

Cualesquiera de estas 3 formas de 'pares' son equivalentes para identificar una misma trayectoria de planeo estable. De entre las trayectorias estables posibles, nos interesan especialmente las de mayor velocidad, o las de mayor autonomía (mínimo gasto energético entre 2 puntos dados).

Normalmente el mayor gasto energético es modificar la masa de lastre variable, por lo cual las estrategias suelen ser mover ese control cuanto menos mejor, y ajustar el resto con la masa móvil longitudinal.

Aprovechamos este punto para hacer una observación sobre esas tres modalidades equivalentes de definir una trayectoria. Algunos de los 6 parámetros que definen los 3 pares pueden medirse con sensores, y otros no. En particular, la posición de la masa móvil y la cantidad de lastre pueden medirse con sensores de posición. También ocurre con el ángulo de trimado y la componente vertical de la velocidad V. Sin embargo, los otros parámetros: ángulo de ataque (alfa) y ángulo de planeo (seta) no podemos medirlos realmente. No tenemos sensores adecuados para ello. La única forma que tenemos de medirlos es indirectamente: a partir otras magnitudes que sí podamos medir con sensores, y completando esa información con las relaciones matemáticas incluidas en el modelo.

3.4 Elección del Modelo 2D como mejor alternativa a nuestro alcance. Puesta en práctica

Después de haber estudiado en profundidad la tesis de **[Graver 2005],** y de haber asimilado toda su complejidad, y después de no haber encontrado ninguna forma fácil de aplicar ese conocimiento a nuestro caso, nos veíamos inevitablemente abocados a utilizar programas avanzados del tipo MATLAB u OCTAVE, y transcribir todas las ecuaciones matriciales incluidas en las fórmulas 4.31,4.32 .4.33 y siguientes de aquella referencia para poder hacer algún trabajo realista sobre la simulación del Glider,

Pero entonces, cuando abríamos el OCTAVE para empezar a copiar código, se nos ocurrió buscar (entre los ejemplos de MATLAB) algún modelo de sistema en espacio de estados, para poder usar la estructura de aquel archivo como esqueleto-plantilla sobre el que meter nuestras fórmulas para las matrices típicas A y B de la referencia de **[Graver 2005]**.

Y fué así, al entrar en Google con … 'MATLAB modelo estados Glider simulación' ..., cuando nos encontramos por casualidad con 'algún indicio'… que nos permitió llegar hasta **[Singh 2018]** :

En esa referencia está disponible un código para MATLAB que ya incorpora todas las ecuaciones del modelo de **[Graver 2005]**. Esto es justo lo que necesitamos.

En primer lugar, vemos que ese código fuente en MATLAB se ha puesto a disposición pública en 2018. El equipo dirigido por **Yogang Singh del IIT de Madras**, India, ha puesto a disposición pública en OPEN SOURCE un modelo MATLAB que se ajusta totalmente a la descripción de **[Graver 2005].** Además, incluye un modelo simplificado, restringido a navegación en plano vertical 2D.

Parece claro que el código fuente MATLAB usado por Graver, (Universidad de ***Princeton***, USA) con el que trabajó en su tesis de 2005, y en el que probó su modelo e hizo simulaciones de Gliders comerciales conocidos, no había sido puesto nunca a disposición pública.

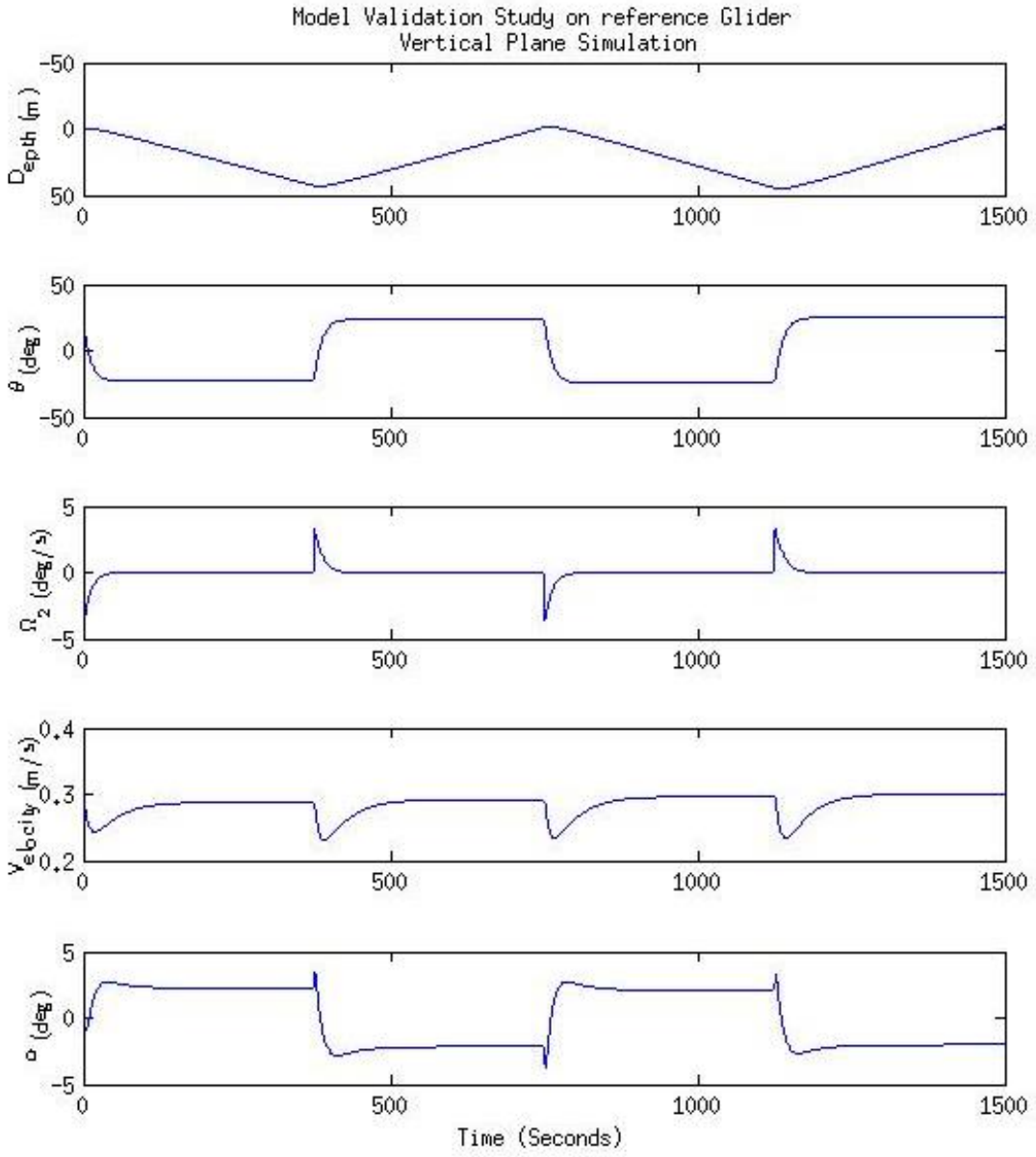
Afortunadamente, doce años después, otra universidad (**Madras**, India) pone en OPEN SOURCE un código MATLAB, que recoge aquellos cálculos. Otro gran paso para la comunidad que defiende la conveniencia del OpenSource. La motivación de los autores **[Singh 2018]** es facilitar la investigación con los diseños de Glider, y el único requisito que imponen es que cuando se use tal code se haga mención a su referencia **[Singh 2018]**, cosa que aquí ya hacemos.

Es muy de agradecer esa aportación, poner a disposición pública el modelo, que hace posible que podamos continuar este trabajo con la calidad que deseábamos.

No habíamos encontrado antes ese code, porque acaba de nacer al acceso público, y apenas se ha hablado de ello en Internet. Desconocíamos su existencia, pero ahora lo hemos encontrado.

3.5 Prueba del modelo de simulación con OCTAVE.

Nos disponemos a probar ese modelo 2D en OCTAVE. El code contiene un ejemplo que permite reproducir la simulación del Slocum realizada en **[Graver 2005]**, y que sirve de prueba de validación. Son 4 planeos sucesivos a ± 30º de trayectoria, de 375 s de duración cada uno.



Esta figura, que hemos elaboración específicamente para este TFM a partir del code de **[Singh 2018]** es nuestra propia ejecución de dicha simulación del SLOCUM.

Después de este primer éxito, nos animamos a revisar el code, comprobando que sigue muy fielmente toda la formulación de **[Graver 2005]**. Con estas pruebas hemos decidido adoptar ese code como válido para nuestro modelo matemático buscado.

4 Determinación de los parámetros de nuestro Glider para incluirlos en el modelo matemático

Algunos de los datos de nuestro modelo podemos deducirlos directamente de su proyecto teórico, y otros, como los relacionados con la hidrodinámica, no.

Para esto últimos tenemos algunas formas de aproximación. Una de las más simples, y creemos que aceptablemente adecuada, es asumir las similitudes entre el Slocum y nuestro Glider, y usar los datos del Slocum, con las conversiones de escala y semejanza debidas, para nuestro Glider. Vamos a considerar la relación de semejanza a partir de las proporciones de las masas de ambos Gliders.

4.1 estimación de parámetros por comparación o semejanza con otros Gliders.

Usaremos la semejanza dimensional para encontrar la relación basada en la escala, entre los parámetros del modelo de referencia y los de nuestro Glider.

El modelo de referencia será el Slocum, ya que existen muchos datos sobre él en la bibliografía.

4.1.1 estimación de momentos de inercia

Las dimensiones del momento de inercia son

Inercia <=> masa . L ^ 2 <= > ro. L ^ 5

con lo que

Inercia1 / Inercia2 <=> (L1 /L2) ^{5} = (Masa1/Masa2) ^ (5/3)

Esto nos dá una relación que nos permite estimar las inercias para nuestro modelo, a partir de la del modelo de referencia, y la relación de masas.

En particular, comparadas con el Slocum (índice1), y nuestro Glider (índice 2)

M1/M2 = 50 / 5 = 10

10^(5/3) = = 45 = Inercia1 / Inercia 2

4.1.2 estimación de la masa añadida.

El valor de masa total incluye la masa de agua atrapada dentro de la carena de las alas y las ojivas, aunque no incluye, en este caso, la masa de agua que puede ser arrastrada por el movimiento del Glider. Esa masa no inlcuida hasta ahora es la que llamamos ‘añadida’.

La masa ‘añadida’ por el movimiento del Glider depende concretamente de la geometría del Glider, y la dirección del movimiento. Tiene alguna relación con el coeficiente de forma y el área de la sección transversal a la dirección del movimiento.

En la referencia **[controlador optimo]** , tabla 2 de aquella referencia, se muestra una comparativa de parámetros de otros Gliders.

Table2: Values assigned to the constant values[14].

Parameters	Vehicle 1	Vehicle - 2	
$g(m/s^2)$	9.81	9.81	
K_{L_0}	0	0	
K_L	306.00	132.5	
K_{D_0}	18.00	2.15	
K_D	109.00	25.00	
K_{M_0}	0	0	
K_M	-36.50	-100.00	
$m(kg)$	11.22	50.00	100
$\overline{m}(kg)$	2.00	9.00	18%
$m_h(kg)$	8.22	40.00	80%
$m_{f_1}(kg)$	2.00	5.00	20
$m_{f_3}(kg)$	14.00	70.00	140%
$J_2(kg \cdot m^2)$	0.10	12.00	

$m_s = m_h + m_b + \overline{m}$ = m+m_o

mb 2% aprox

A efectos informativos esta tabla, sacada de la referencia **[Graver , 2005]** , se refiere a los datos del Slocum.

A la vista de esos datos vamos a estimar las masas añadidas (con subíndices f1 f3 en la tabla) para nuestro Glider. Para ello vamos a asumir algunas hipótesis:

1 que existe semejanza geométrica con aquellos vehículos,

2 que la masa añadida es proporcional a la superficie proyectada por el vehículo en la dirección del movimiento.

Con la segunda hipótesis: la masa añadida a considerar será distinta según el eje del movimiento afectado. Es decir: mf3 es la masa añadida considerada en un movimiento según el eje 3 del vehículo 1, es decir perpendicular a la vista en planta. y mf1 es la masa añadida considerada en un movimiento a lo largo del eje principal del cuerpo del Glider (hacia proa)

Con la primera hipótesis: Entre cuerpos semejantes con una relación de escala lineal BETA podemos decir

$L1/L2 = beta$

$Area1/area2 = beta^2$

$Masa1/masa2 = Vol1/Vol2 = beta^3$

Estas masas que estamos relacionando son las del volumen de desplazamiento.

- Con la 1ª hipótesis: si comparamos nuestro Glider (lo llamaremos Vehículo3), de masa3= 5 kg, con el vehículo1 de la tabla, masa1 = 11.22 kg, tendremos

  $masa1/masa3 = 11.22/4.8 = 2.33 = BETA^3 \rightarrow BETA = 1.33$

- Con la hipótesis 2ª: masaf1_j = kj.area1, j = 1 ,2, 3 componentes axiales

  $area1/area3 = BETA^2 = 1.76$

  $area1/area3 = masaf1\_j/masaf3\_j$

con lo que las masas añadidas para nuestro Glider de 4.8 kg en el cuerpo, serán

$masaf3\_j = masaf1\_j/BETA^2$
j1 , $masaf3_1 = 2/1.76 = 1.14$ k
j3 $masaf3_3 = 14/1.76 = 7.9$ k

los subíndices j1 y j3 se refieren a las componentes 1 y 3 del vector genérico 3D (x y z)

4.1.3 Estimación de coeficientes hidrodinámicos

Las fuerzas **hidrodinámicas** generan la resistencia y la sustentación.

Además, sabemos por la teoría de alas finitas que

[Fuerzas hidrodin] = (½ . ro . S . $(X\_p)^2$) * [CL CD]

esas fuerzas son perpendiculares, y una de ellas está en línea con la velocidad X_p (vector).

En las ecuaciones usadas en [**Singh**] y [**Graver**], los coeficientes hidrodinámicos no son adimensionales. Es una forma poco común en el ámbito de la hidrodinámica, pero es el que se ha usado en esas fuentes.

Así, esos coeficientes describen relaciones entre Fuerzas y Momentos sobre un móvil con una superficie característica A, (área), y velocidad V, en la forma

Fuerza, Momento … = 1/2 ro.A. V^2 x Cfuerza,...CMom

= K V^2

y las dimensiones de esos coeficientes K serán las de (Fuerza,..Momento)/V^2

es decir:

siendo

ro dimension de densidad : M / L3

M dimensión Masa

L dimensión Longitud

T dimensión Tiempo

para fuerzas:

M L /T2 / (L2 / T2) = M / L, pero M = ro . L3, …. luego

$= ro. L2$

$= ro. L^2$ dimensiones para Kfuerza

y para momentos:

$M\ L\ L/T2\ /\ (L2/\ T2) = M$

$= ro. L3$

$= ro. L^3$ dimensiones para Kmomento

Con lo cual, las relaciones entre los coeficientes de modelos dados estarán en proporción a los cocientes de las potencias 2 y 3 de las magnitudes lineales representativas de su tamaño.

En la práctica, entre modelos semejantes ocurre

$Masa1 /Masa2 = Volumen1/Volumen2 = (L1/L2) ^ 3$

y por tanto, $(L1/L2) ^ 2 = (Masa1/Masa2) ^ (⅔)$

y $(L1/L2) ^ 3 = (Masa1/Masa2)$

Con lo cual ya sabemos qué relación hay entre los coeficientes hidrodinámicos

- de Fuerzas $Kf1/Kf2 = (Masa1/Masa2) ^ (⅔)$
- y de momentos $Km1/Km2 = (Masa1/Masa2)$

Además de lo anterior, para compensar que nuestras alas tienen un perfil tipo NACA 0020, y el de la referencia **[SLOCUM]** es NACA 0012, hemos penalizado los

coeficientes con unos factores correctores (Menos sustentación y más resistencia de arrastre):

- para KL, factor 0.85, menor que 1, representa un 85%
- para KD, factor 1.5 y 1.1, para los dos términos de la función de alfa

Con todo ello esperamos haber deducido con buen fundamento los parámetros que definen el comportamiento cualitativo y cuantitativo de nuestro modelo de Glider.

4.2 Otra forma de determinar los parámetros del Glider. Identificación de parámetros. Caso de un Glider existente que podamos ensayar.

Cuando tenemos un Glider construido, y podemos usarlo, es posible obtener experimentalmente datos para determinar los parámetros del modelo matemático que lo representa. Es lo que se denomina *Identificación de Parámetros*.

Entre los diversos métodos de ensayo para hacerlo, uno relativamente sencillo y conocido (al menos en los buques de superficie) es la experiencia de estabilidad. Para los sumergibles también hay una modalidad de experiencia de estabilidad que nos permite identificar algunos de sus parámetros

4.2.1 una Experiencia de Estabilidad para el Glider

En nuestro modelo matemático podemos considerar una situación de velocidad de traslación nula, o mínima, en la que las fuerzas hidrodinámicas son despreciables. En esas condiciones, la dinámica se reduce a oscilaciones como un péndulo, dentro del plano longitudinal del Glider, que es vertical. Ese movimiento tiene por modelo la EDO de trimado, una EDO de segundo orden muy conocida y estudiada por ser común a muchos fenómenos físicos.

En un ensayo de estabilidad, con el Glider en el agua, vamos a poder determinar parámetros relevantes, característicos del trimado y del movimiento de cabeceo. En lo que sigue vamos a explicar ese ensayo de estabilidad.

Moveremos los controles de flotabilidad y de posición de masa móvil hasta encontrar una posición en equilibrio, es decir:

- flotabilidad neutra: flotabilidad=0. Nos fijaremos en la posición de émbolo que lo hace posible
- trimado nulo: trimado =0. Nos fijaremos en la la posición de la masa móvil que lo hace posible

Ambas condiciones deben cumplirse simultáneamente. Si el recorrido de actuación del émbolo no permite alcanzar una situación de flotabilidad nula, deberá modificarse el lastre sólido externo hasta lograrlo. Lo mismo respecto a la condición de trimado nulo, asiento =0, y la posición de la masa móvil.

Los dos parámetros (Masa de lastre o posición de émbolo de lastre, y posición de la masa móvil) así obtenidos, y medidos, hacen que cuando los actuadores estén en esa posición, el Glider quedará parado y con asiento nulo. Si previamente tenía velocidad, esta se reducirá hasta pararse.

Partiendo de la situación de equilibrio anterior, movemos la masa móvil una cantidad determinada, y comprobamos el cambio de asiento (teta).

Esperamos a que se quede quieto y eso supone que las derivadas primera y segunda de teta respecto al tiempo son nulas:

teta_p = 0; teta_pp =0

eso deja la EDO del movimiento de balance reducida a

k_restauracion elástica * teta = Mom_exterior

k_restauracion elastico = momento masamovil/teta

k_restauracion elastico = masatotal *BG

Conocemos el momento aplicado por mover la masa móvil una cantidad concreta, y además conocemos el ángulo teta logrado.

por tanto conocemos el producto masa_total * BG

La separación de los dos factores del producto, masa_total y BG, requiere conocer alguno de ellos, u otra relación. Nosotros podremos calcular la masa total. Determinaremos la masa aplicando el principio de Arquímedes, igualándola a la masa del volumen (calculado por integración de las formas) de líquido desplazado. Esa situación de igualdad se produce durante el ensayo cuando, estando el Glider totalmente sumergido, no hay tendencia clara en hundirse o flotar,

En los sumergibles previstos para ocupación humana, esta es la experiencia de estabilidad que se hace realmente, y el valor del volumen total es conocido mediante cálculo previo sobre plano. Para asegurar que la masa del glider coincide con la del liquido desplazado, o al menos la diferencia es muy pequeña, se dispone un sistema para acotar con precisión el máximo de esa diferencia. Para medir la masa del volumen de agua desplazado se hace un ensayo consistente en comparar con el peso aparente (en el agua) del sumergible. Para ello se dispone una pequeña boya (en forma de cilindro vertical) amarrada al sumergible con un cabo inextensible de longitud suficiente (pero no grande) para que la boya pueda flotar con el sumergible totalmente sumergido, y se lastra o deslastra el sumergible hasta que la flotabilidad negativa del Glider existe, pero no es suficiente para hundir totalmente la boya. Es decir, el cabo toma tensión, pero la boya permanece interceptando la superficie libre del líquido. El volumen de la fracción sumergida de la boya es la pequeña diferencia que existe entre la masa del glider y la del volumen de agua desplazada. Normalmente es un porcentaje (entre 0.1 y 0.5%). Cuando las oscilaciones

arriba y abajo del sumergible (y la boya) se reducen lo suficiente, significa que estamos muy próximos al punto de equilibrio.

Hay que tener presente que los movimientos son muy lentos, pues las fuerzas que se producen son del orden del volumen sumergido de la boya, y en cambio, la masa que debe moverse es la de todo el submarino. Por ejemplo, en un caso realista un sumergible cilíndrico de unos 15 m de eslora y 3 m de diámetros desplaza unas 30 toneladas, y la boya que se usa para el ensayo es de unos 25 litros. Lo que significa que la precisión con que se ajusta la hipotética igualdad entre la masa y el desplazamiento es de 25/30000 == 0.1% . La bomba de lastre deberá hacer correcciones del orden de unos pocos litros hasta ajustar el punto de equilibrio.

En un ensayo real de un sumergible bien diseñado no conocemos con total precisión el volumen 'real' de desplazamiento, pues existen los 'apéndices'. Los apéndices son elementos de volumen relativamente pequeño, que no se consideran en el cálculo del denominado volumen 'de trazado'. La razón de su exclusión es que complican el cálculo y su aportación a la suma total es poco relevante. Pueden ser volúmenes positivos o negativos (a restar o añadir al de trazado para obtener el hipotético volumen real de desplazamiento). En algunos casos se ha estimado que el volumen de los apéndices es del orden del 1-3 % del volumen de trazado. El agua ocupa, o es evacuada de, esos espacios 'apéndices', pero en el cálculo no se tiene en cuenta.

Tampoco conocemos con total precisión la masa del sumergible. Lo que sí logramos durante el ensayo realizado de esta forma, con la boya, es que la diferencia entre ambos (masa del glider y masa de desplazamiento) es conocida y su valor es la masa desplazada por el volumen sumergido de la boya.

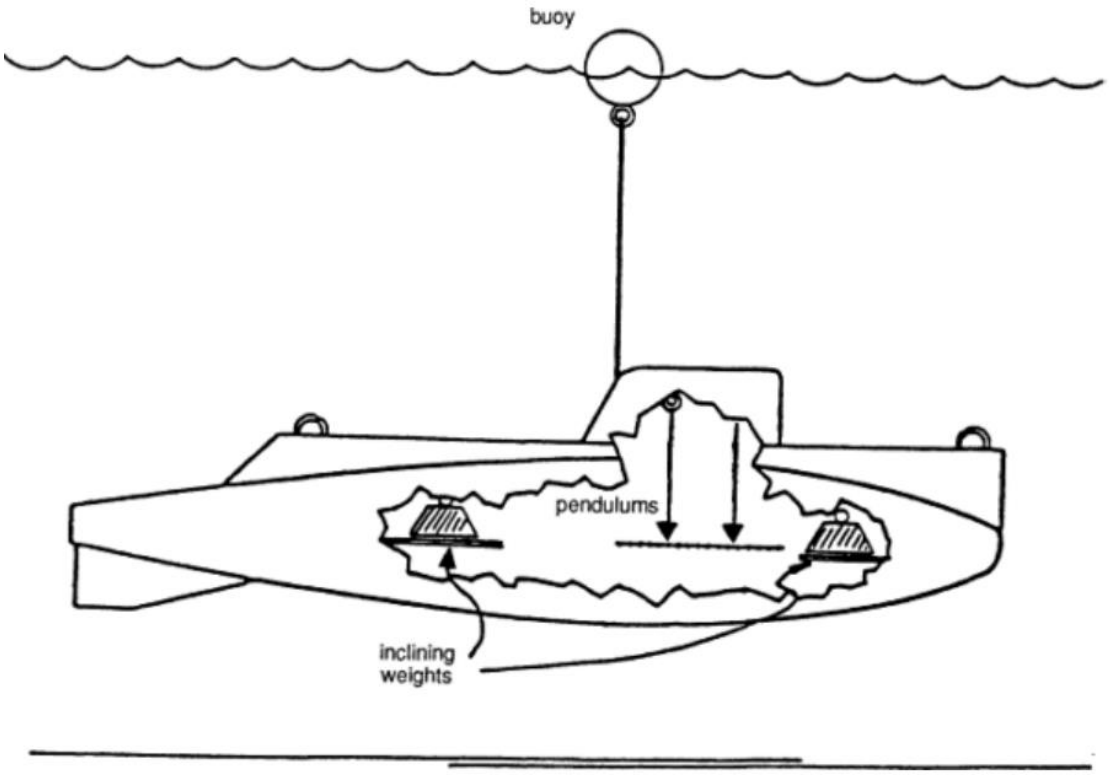


Figure 5 Inclining Experiment

la presente figura, sacada de la pag 84 de la referencia **[Safety of TOURIST SUMERGIBLES]** muestra la disposición durante un ensayo de estabilidad en el caso de un sumergible turístico de pasaje.

Cuando se consigue el equilibrio, la diferencia entre la masa del sumergible (aún no conocida) y la del volumen de agua desplazada (asimilado al volumen de trazado, calculado) es justamente la masa del volumen de la carena de la boya semisumergida (medible en el ensayo).

Además de deducir la masa del sumergible, podremos deducir BG, que es el otro factor que determinaba un producto conocido.

En el ensayo-experiencia de estabilidad, también se debe medir el periodo de balance 'natural' del sistema oscilante, y si es posible medir varias oscilaciones

que permitan ver el grado de amortiguamiento de las oscilaciones. Con ello podremos deducir los coeficientes de teta_pp y teta_p en el EDO de balance.

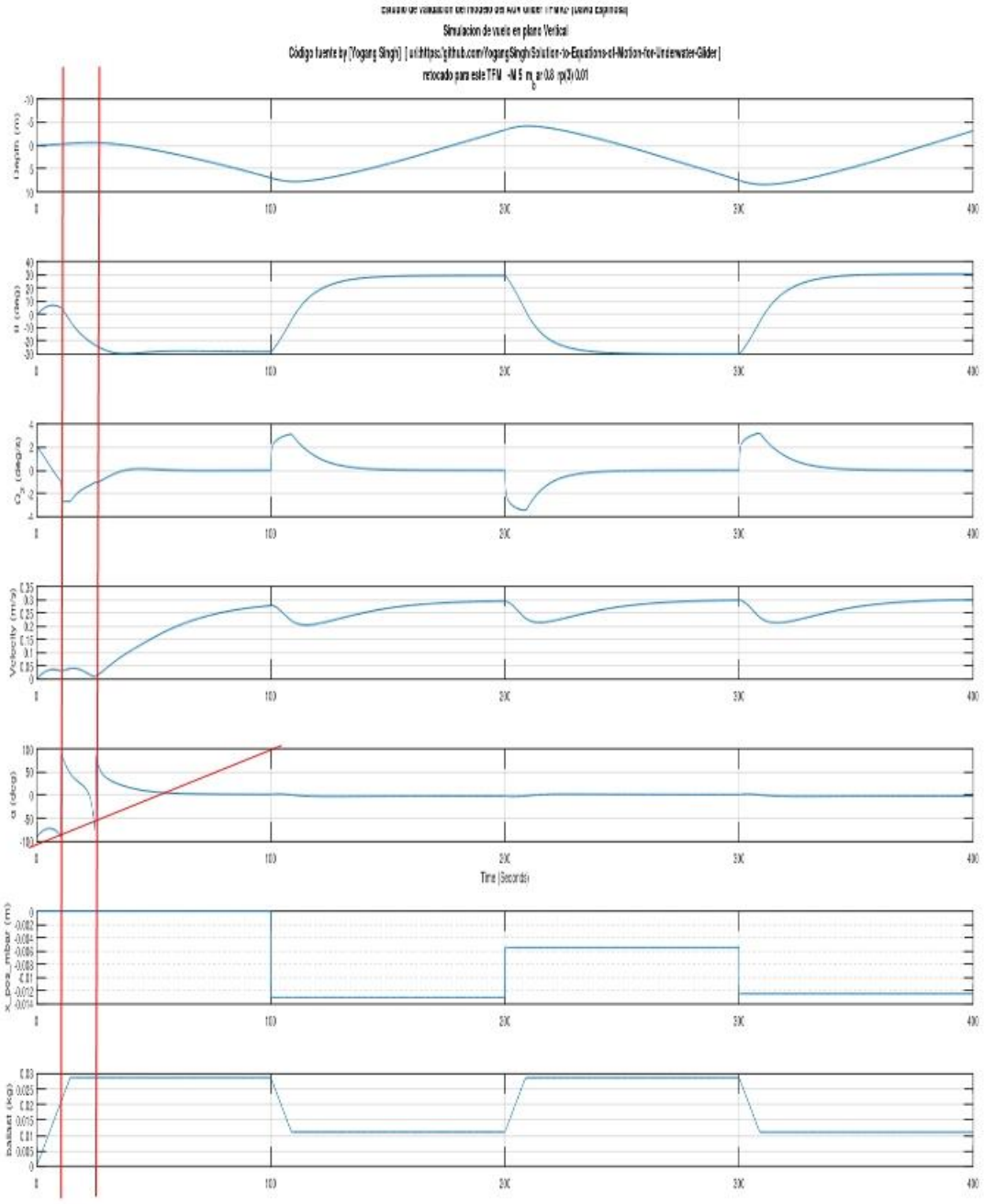
El estudio teórico de la EDO de segundo orden, lineal, del péndulo físico con amortiguamiento es un caso clásico en muchos tratados de física, por lo que todas las relaciones entre periodo, amortiguamiento de la amplitud de oscilación, y coeficientes de la EDO están tratadas en la literatura.

Las inercias y masas que aparecerían en ese estudio empírico con movimiento serían las del Glider junto con la del agua 'añadida', que acompaña el movimiento del cuerpo del Glider.

Esa masa de agua 'añadida' tiene una parte realista y otra ficticia. El conjunto da explicación a que una cantidad (difícil de concretar) de agua se mueve 'siguiendo a', o 'siendo arrastrada por' el movimiento del Glider .

Teniendo en cuenta esas masas e inercias 'añadidas', se consigue una mejor coincidencia entre las mediciones reales y las que predice el modelo (linealizado y simplificado) de la EDO.

5 Simulación de navegación. Comportamiento de nuestro Glider con los parámetros que hemos estimado.



La figura anexa muestra una simulación:

Es la evolución a lo largo del tiempo (400s, = 4 x 100 s carreras) de los parámetros siguientes:

1. profundidad,
2. asiento (º),
3. velocidad de cabeceo,
4. velocidad lineal de traslación,
5. alfa (ángulo de ataque º),
6. x_bar (abscisa o posición de la masa móvil),
7. Masa de lastre contenida en la cámara de flotabilidad de proa.

La navegación prevista era a una velocidad máxima V = 0.3 m/s, con un ángulo de planeo de 30º (en descenso y en ascenso), y no se ha fijado una profundidad máxima y mínima en la que hacer la inversión del movimiento, sino que esa inversión se ha mandado hacer, según un temporizador, cada 100 s,.

La masa del Glider es de 5 kg, el parámetro de estabilidad es BG = 5 mm, la masa móvil es m_bar = 0.800 kg

5.1 Comentario a la navegación simulada en nuestro GLIDER

Después de cargar los parámetros de nuestro Glider en el modelo matemático, hemos probado algunas simulaciones, como la que se muestra en la figura adjunta. En ella se aprecia con mucha claridad que el inicio de la inmersión es accidentado.

Se empieza con trimado cero y con una leve flotabilidad (ascendente), pero la masa móvil (longitudinalmente) está demasiado hacia popa (realmente justo en el centro) y dificulta el trabajo de hundir la proa. En su lugar crea el giro inverso, apopante, hasta los teta 7º (en T5s). Ese giro solo se detiene e invierte por la propia hidrodinámica, que hace que el borde de salida de las alas se comporte como borde de ataque, y por el lastrado de la proa, que avanzando en el tiempo hace la proa más pesada y capaz de tirar hacia abajo de todo el Glider.

Narrando la gráfica en detalle:

1. Comienza (T 0s) con trimado teta = 0º, y una la leve flotabilidad remanente que existía desde el instante (T=0). Eso hace que el Glider suba, y a la vez va apopando. Durante ese tiempo recibe el flujo aparente del agua por encima, con alfa -75º, en (T 5s).
2. En (T 6s) deja de apopar (teta ha llegado a 8º) y comienza a invertir el giro. Todavía sigue subiendo.
3. La flotabilidad se anula en (T=10s), pero por la inercia todavía sigue subiendo.
4. En (T 12.5) pasa por trimado nulo, teta =0, el ángulo de ataque alfa pasa de negativo - 89 a positivo + 89, y continua reduciéndose a medida que el asiento se hace más negativo (proa abajo).

5. En (T 14s) el lastrado termina, pero por inercia continúan los movimientos de ascenso, y de cambio de trimado (teta = -8º), cada vez más negativo
6. En (T=25s) deja de subir, pasa por velocidad 0, y empieza a bajar.
7. El trimado es teta = -24º, y casi de inmediato el ángulo de ataque alfa pasa de negativo -88 a positivo alfa =+89 y continuamente evoluciona hacia alfa = +5º
8. En (T 50s) teta ya ha dejado de oscilar, y está estabilizado en teta = -28º. La velocidad sigue aumentando lentamente, y el ángulo de ataque va reduciéndose lentamente hacia el valor alfa = 3.5º. Salvo la velocidad ya todo está prácticamente estabilizado y se mantiene constante. El ángulo de planeo es de 30º, como se pretendía.
9. El resto de la carrera hasta (T=100 s) se mantendrá todo igual. Se alcanza los 7 m de profundidad.
10. En ese momento se produce la orden de cambio de dirección: Navegación Arriba Ascendiendo. Eso genera unos cambios de los cuales el primero y más acusado es el traslado de la masa móvil de 0.800 kg, hacia popa, moviéndose unos 7 mm, y creando un cambio de asiento a ritmo constante de 2º/s.
11. Además, el proceso de deslastrado que se inició en (T=100s) ha ido generando menos pesadez y desde el instante (T=108s) ya se inicia el movimiento ascendente, con el asiento =0 y apopando, y pasando por el mínimo de velocidad relativa de la maniobra. Durante esos 8 segundos se ha producido un movimiento muy condicionado por la inercia, que ha

producido una transición de navegación suave, pese a las relativamente violentas acciones de control:

- paso casi inmediato (1 s) de la masa móvil de avante a atrás (unos 10 mm),
- deslastrado total en 7 s.

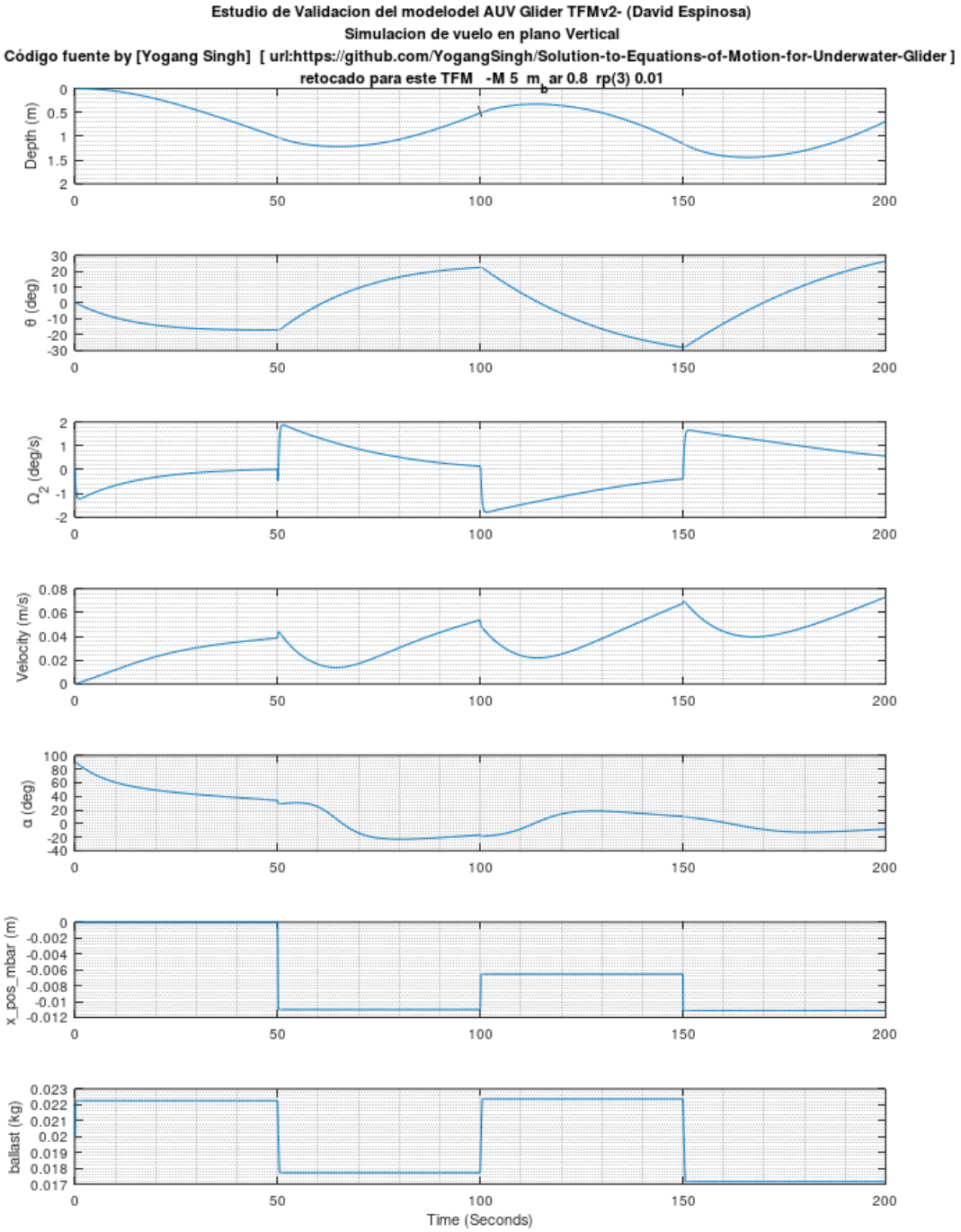
12. A los 8 segundos del inicio de la maniobra de cambio hacia navegación ascendente, el proceso se ha completado, y el Glider está acelerando desde un 75% de la velocidad (máxima) que tenía al inicio de la maniobra. Esta segunda maniobra ha sido mucho más limpia que la de inicio (en T 0s).

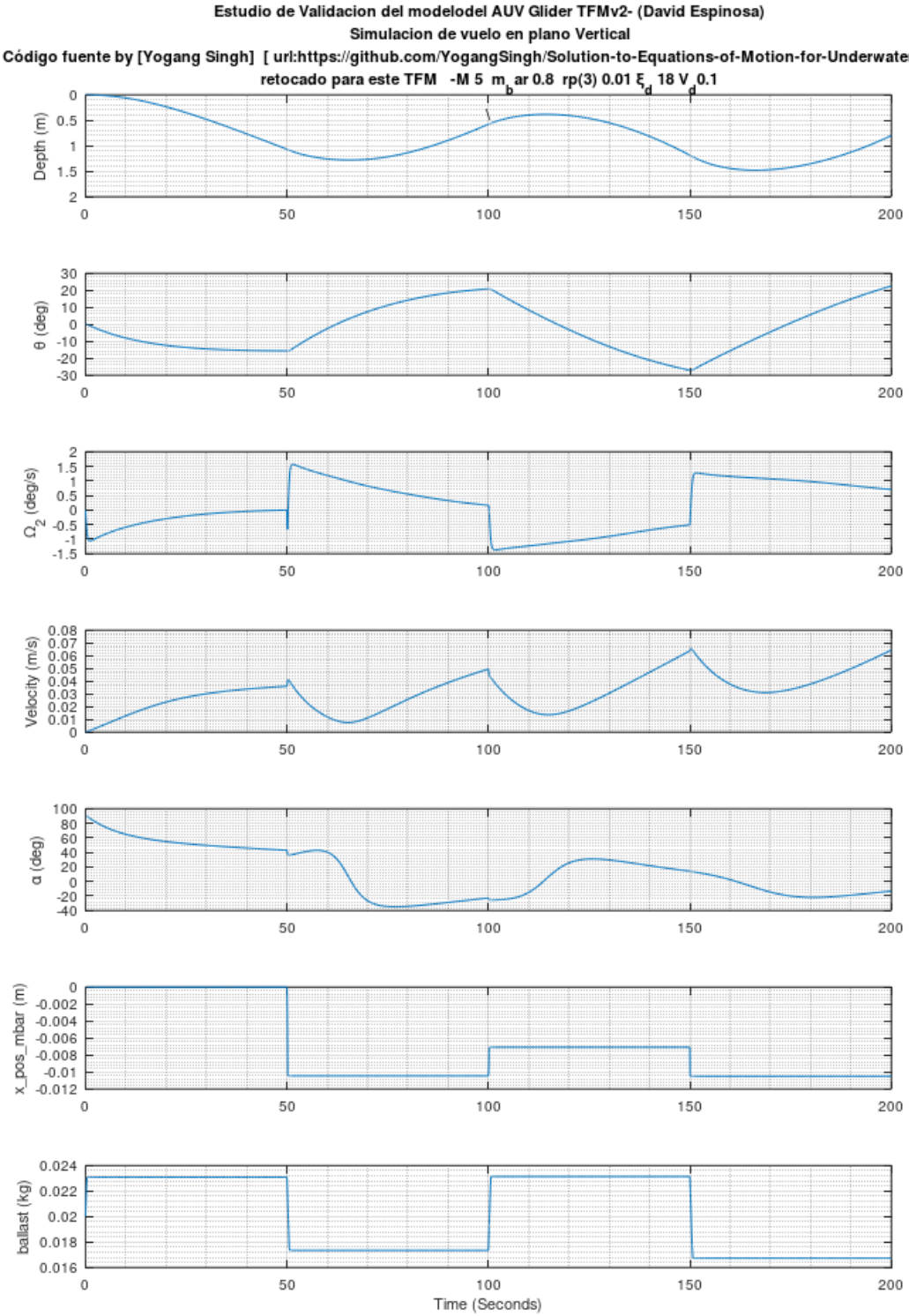
Una de las claves para que las maniobras se realicen bién, con transiciones suaves y con poca pérdida de energía, es que los valores requeridos para las posiciones de la masa móvil y cantidad de lastre se conozcan perfectamente antes de que se inicie la maniobra. Si, además, las acciones de control se ejecutan con precisión y en un tiempo relativamente pequeño, la inercia de velocidad mantiene todo fluyendo correctamente. Además, partíamos de un Glider lanzado a buena velocidad, y por ello, con fuerzas hidrodinámicas (sustentación) muy apreciables, que han ayudado a la maniobra.

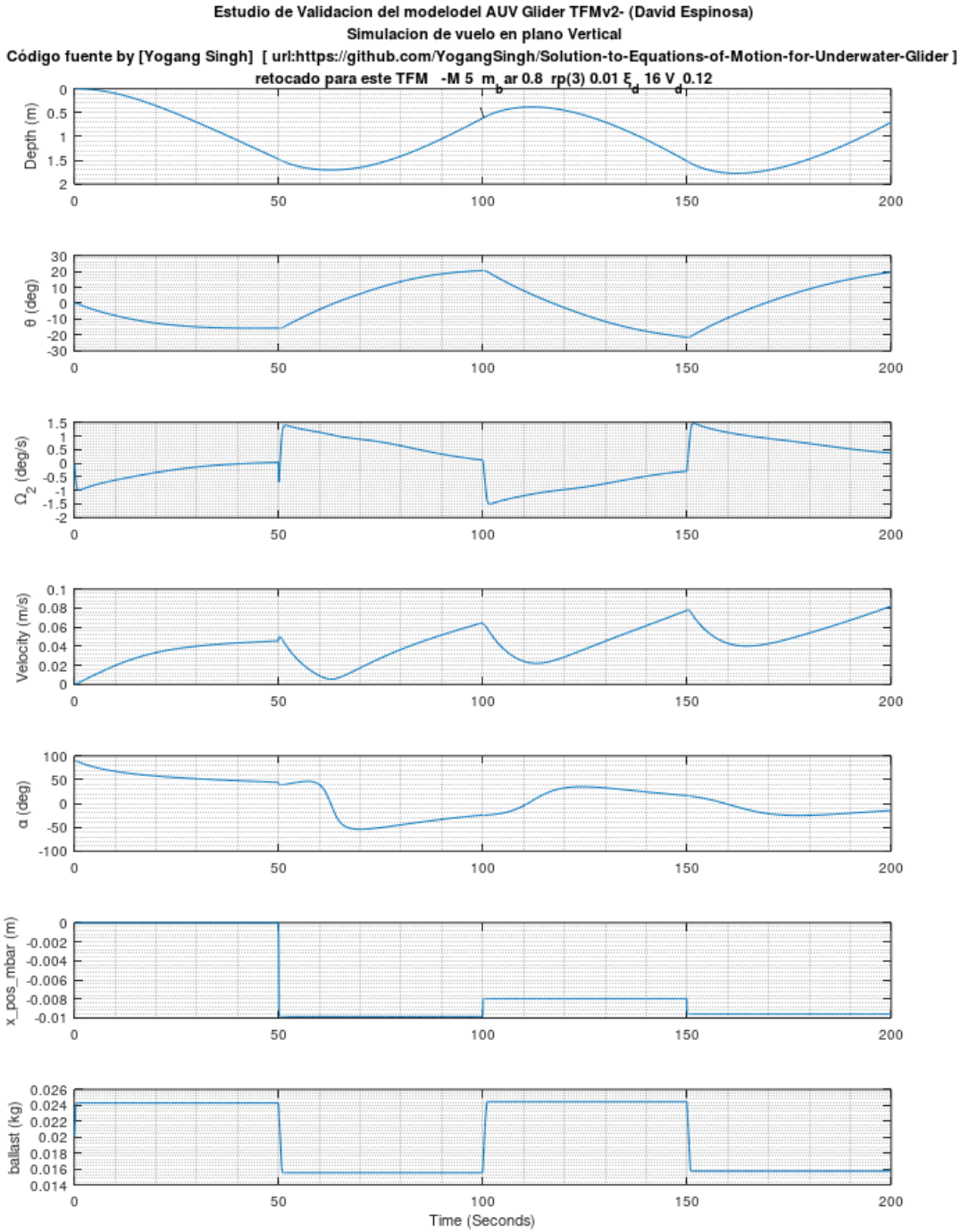
Como en una navegación real, la calidad de la maniobra depende de la eficacia de las acciones de control. En nuestro caso los tiempos de respuesta dependerán de la fuerza de los motores y servos.

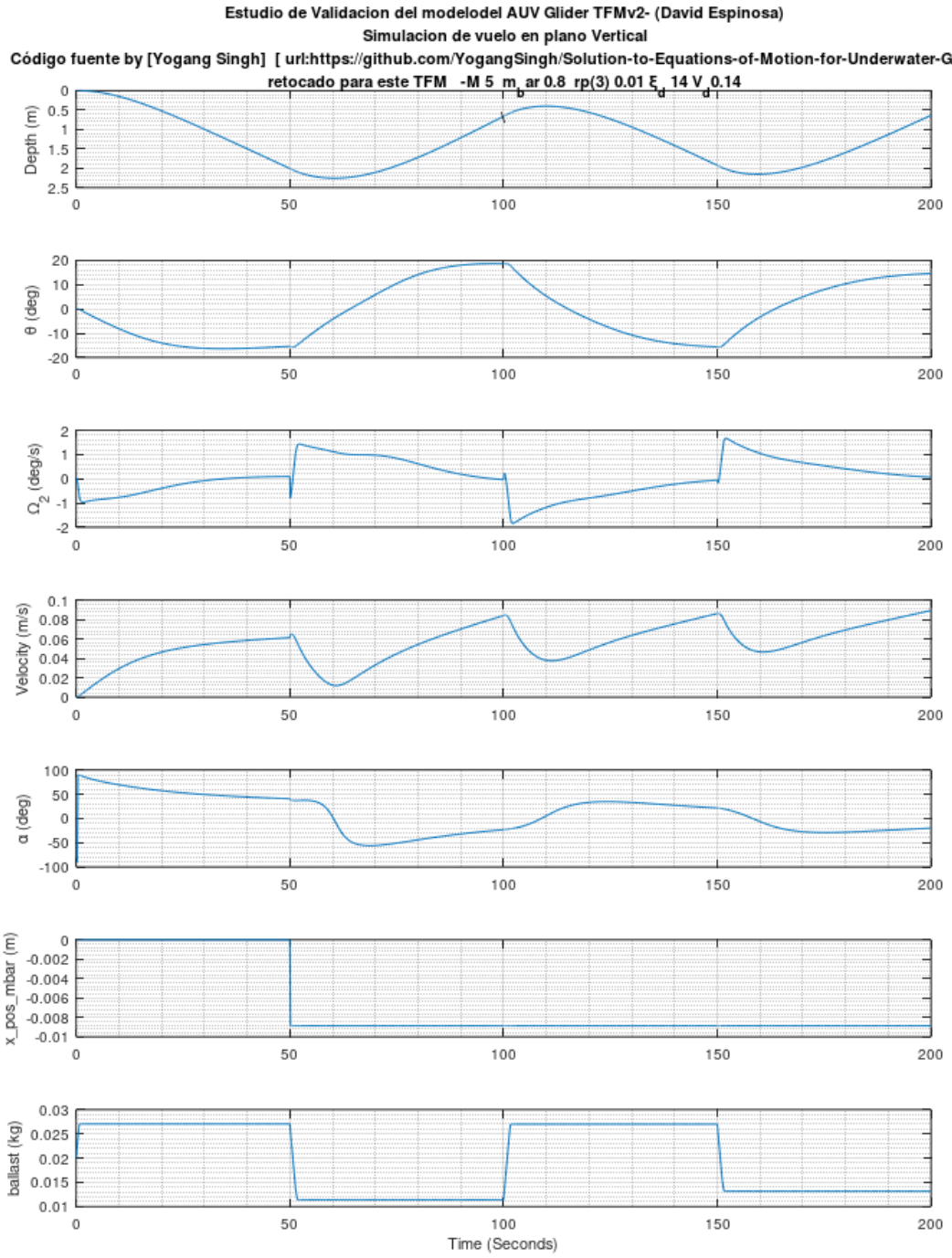
5.2 Otros vuelos simulados de nuestro Glider

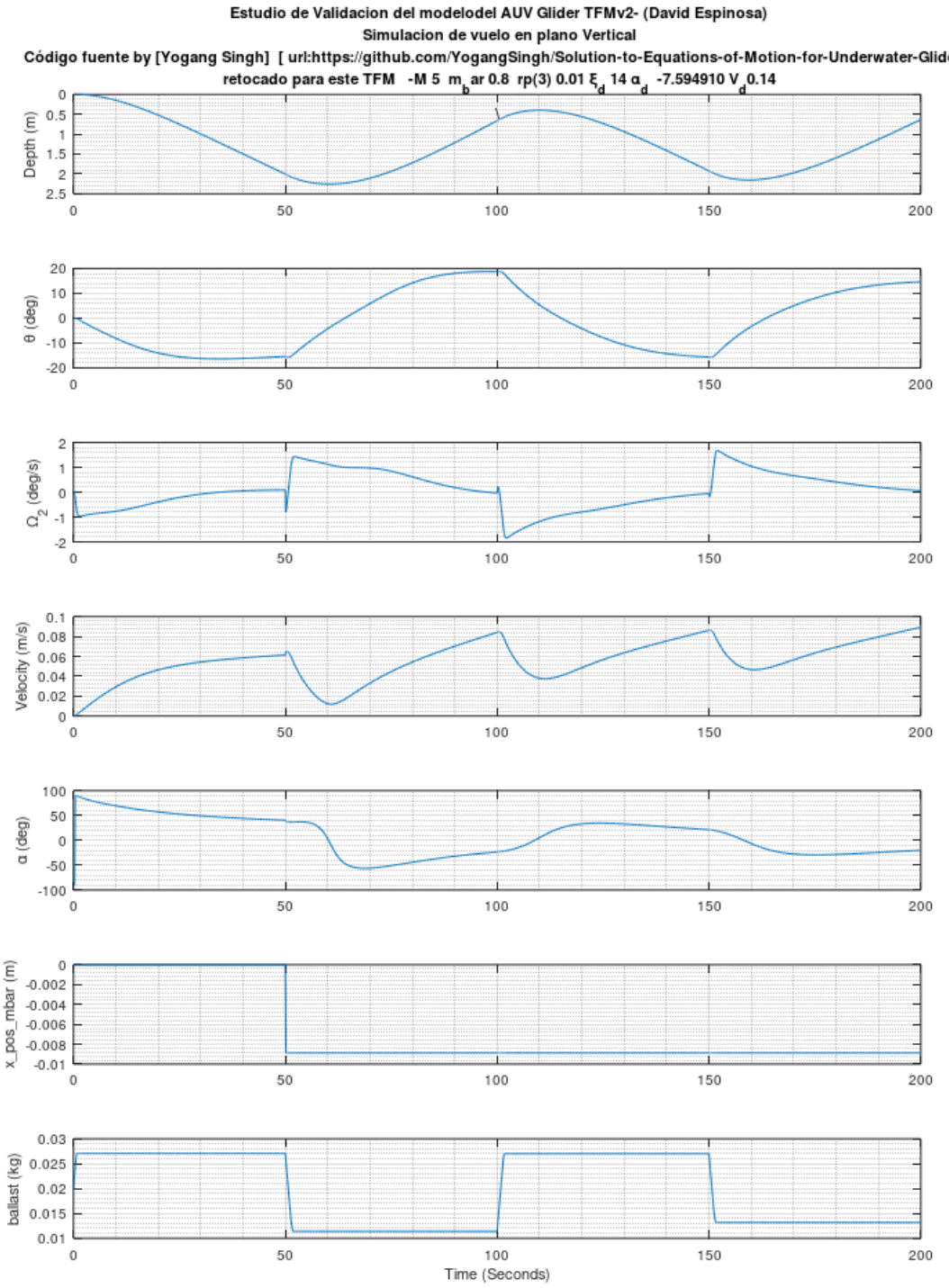
Senda de planeo, Xi 20 º; velocidad máxima , V 0.1 m/s

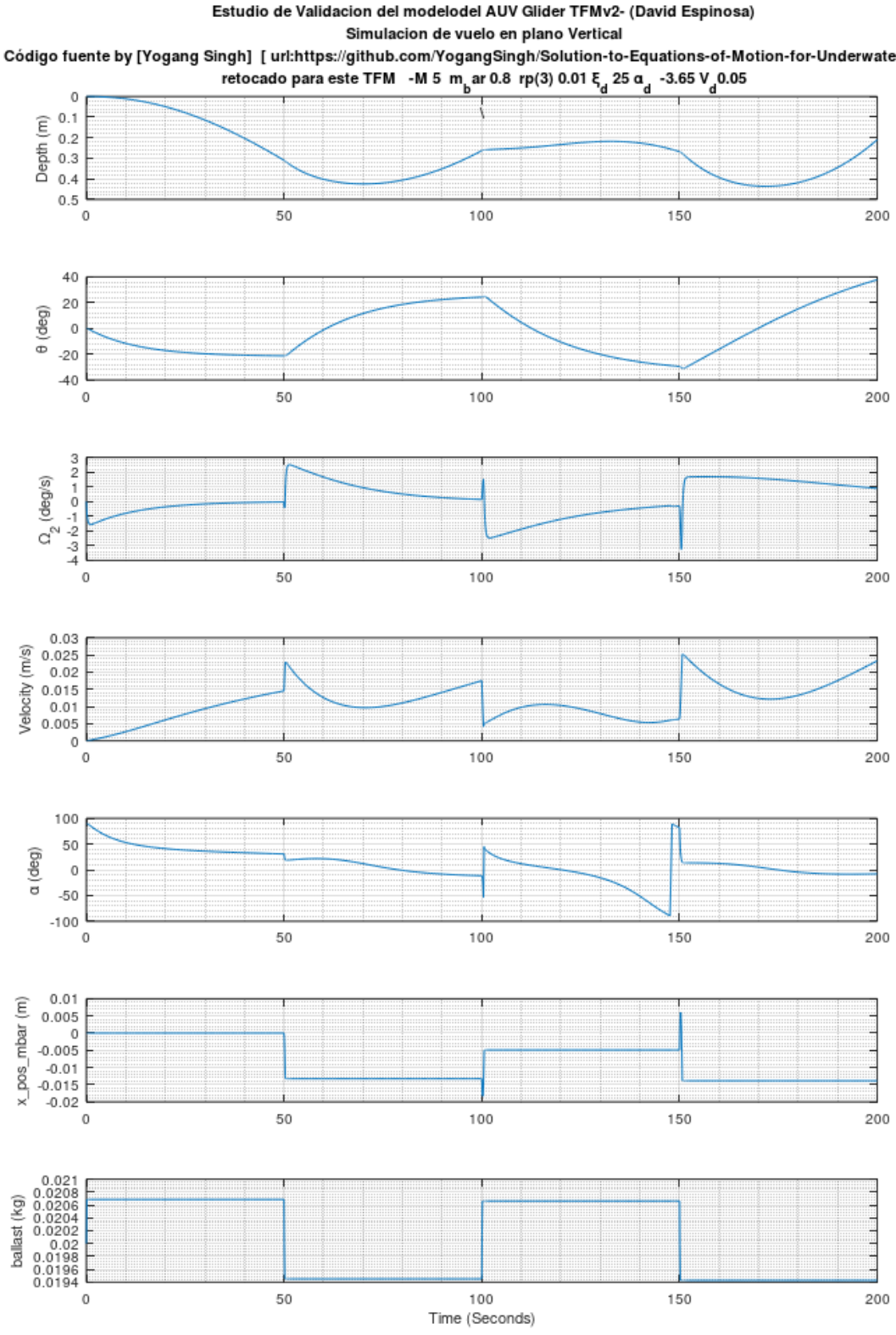


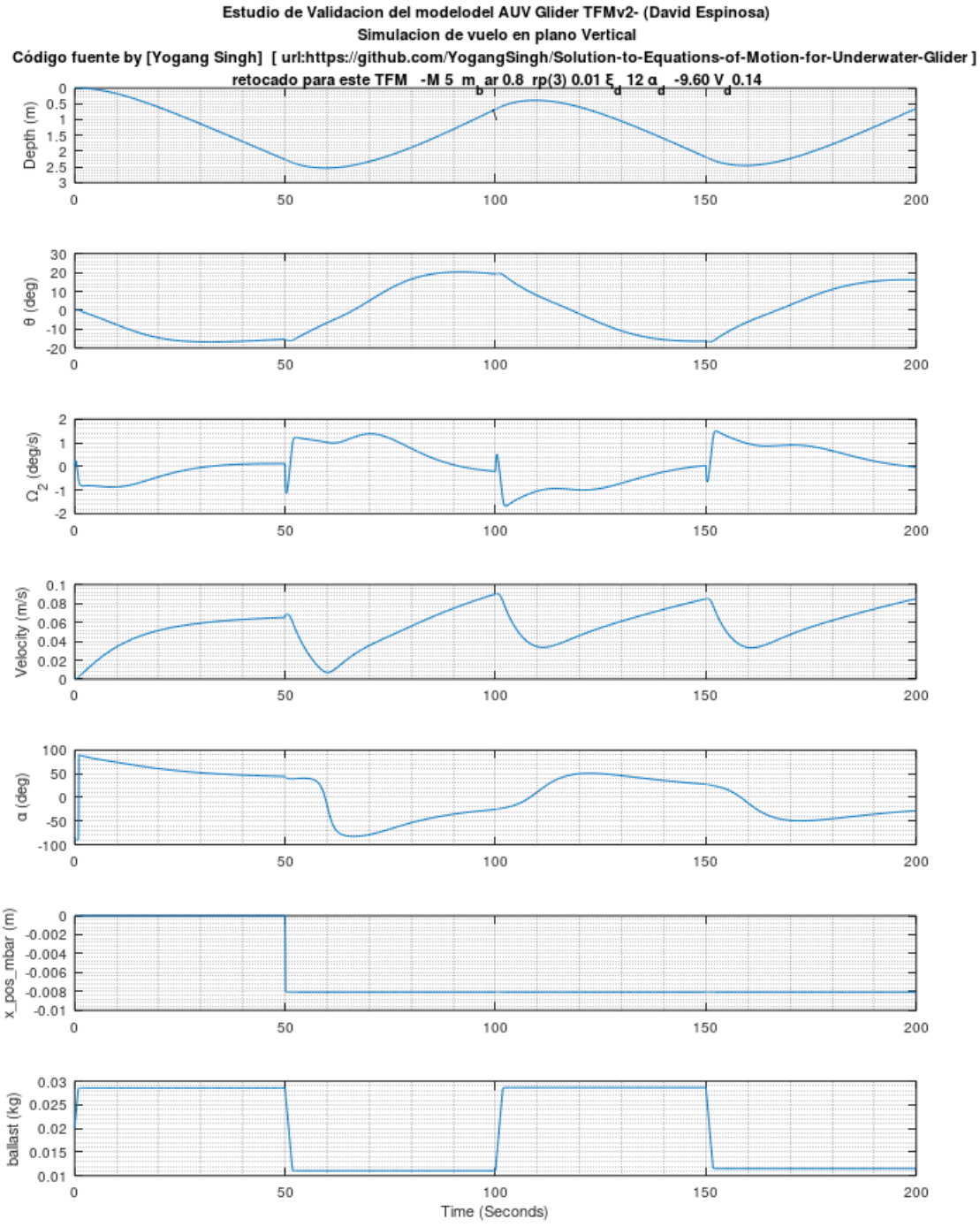

Estudio de Validacion del modelodel AUV Glider TFMv2- (David Espinosa)
Simulacion de vuelo en plano Vertical
Código fuente by [Yogang Singh] [url:https://github.com/YogangSingh/Solution-to-Equations-of-Motion-for-Underwater-Glider]
retocado para este TFM -M 5 m_b ar 0.8 rp(3) 0.01 ξ_d 18 V_d 0.1
Depth (m)
θ (deg)
Ω2 (deg/s)
Velocity (m/s)
α (deg)
x_pos_mbar (m)
ballast (kg)
Time (Seconds)


Estudio de Validacion del modelodel AUV Glider TFMv2- (David Espinosa)
Simulacion de vuelo en plano Vertical
Código fuente by [Yogang Singh] [url:https://github.com/YogangSingh/Solution-to-Equations-of-Motion-for-Underwater-Glider]
retocado para este TFM -M 5 m_b ar 0.8 rp(3) 0.01 ξ_d 16 V_d 0.12
Depth (m)
θ (deg)
Ω2 (deg/s)
Velocity (m/s)
α (deg)
x_pos_mbar (m)
ballast (kg)
Time (Seconds)


Estudio de Validacion del modelodel AUV Glider TFMv2- (David Espinosa)
Simulacion de vuelo en plano Vertical
Código fuente by [Yogang Singh] [url:https://github.com/YogangSingh/Solution-to-Equations-of-Motion-for-Underwater-Glider]
retocado para este TFM -M 5 m_b ar 0.8 rp(3) 0.01 ξ_d 14 V_d 0.14
Depth (m)
θ (deg)
Ω2 (deg/s)
Velocity (m/s)
α (deg)
x_pos_mbar (m)
ballast (kg)
Time (Seconds)


Estudio de Validacion del modelodel AUV Glider TFMv2- (David Espinosa)
Simulacion de vuelo en plano Vertical
Código fuente by [Yogang Singh] [url:https://github.com/YogangSingh/Solution-to-Equations-of-Motion-for-Underwater-Glider]
retocado para este TFM -M 5 m_b ar 0.8 rp(3) 0.01 ξ_d 14 α_d -7.594910 V_d 0.14
Depth (m)
θ (deg)
Ω_2 (deg/s)
Velocity (m/s)
α (deg)
x_pos_mbar (m)
ballast (kg)
Time (Seconds)


Estudio de Validacion del modelodel AUV Glider TFMv2- (David Espinosa)
Simulacion de vuelo en plano Vertical
Código fuente by [Yogang Singh] [url:https://github.com/YogangSingh/Solution-to-Equations-of-Motion-for-Underwater-Glider]
retocado para este TFM -M 5 m_b ar 0.8 rp(3) 0.01 ξ_d 25 α_d -3.65 V_d 0.05
Depth (m)
θ (deg)
Ω2 (deg/s)
Velocity (m/s)
α (deg)
x_pos_mbar (m)
ballast (kg)
Time (Seconds)


Estudio de Validacion del modelodel AUV Glider TFMv2- (David Espinosa)
Simulacion de vuelo en plano Vertical
Código fuente by [Yogang Singh] [url:https://github.com/YogangSingh/Solution-to-Equations-of-Motion-for-Underwater-Glider]
retocado para este TFM -M 5 m_b ar 0.8 rp(3) 0.01 ξ_d 12 α_d -9.60 V_d 0.14
Depth (m)
θ (deg)
Ω2 (deg/s)
Velocity (m/s)
α (deg)
x_pos_mbar (m)
ballast (kg)
Time (Seconds)

Esta última gráfica (senda de planeo 25º, V = 0.050 m/s), es un buen ejemplo de algunas irregularidades aparentes en el comportamiento de la simulación.

En cada transición, en los t 50, 100, 150, se comprueba que variables como V tienen saltos instantáneos que no pueden ser reales. Se deben a que a su vez los saltos de Ballast, lastre y x pos_mbar también son demasiado bruscos (instantáneos, irreales).

Tampoco Omega debe tener saltos bruscos, ni retrocesos instantáneos puntuales. Todo ello se puede explicar como soluciones matemáticas irreales. Son cambios instantáneos, imposibles en el mundo físico real.

La solución para este software de simulación es reducir la velocidad de lastrado, y de movimiento de la masa móvil: reducir su aceleración y su velocidad. y llevarlos a valores razonables, compatibles con los componentes electromecánicos reales que vayamos a instalar.

Podemos resumir esos extraños comportamientos en que el software, que simula cuánto y cómo mover la masa móvil y la cantidad de lastre que debe tener la cámara de flotabilidad y el tiempo en que se debe lograr, no está bién programado. Son criterio simples de tipo *If...then*, pero tienen algunos defectos de coherencia y oportunidad. Realmente la simulación (el comportamiento de la EDO) es correcta y reacciona de esas formas 'raras', cuando los elementos de control, masa móvil y masa de lastre, se manejan incorrectamente (brusco, instantáneo, etc.)

5.3 Valores de consigna para determinar un vuelo uniforme.

Los parámetros necesarios para caracterizar un vuelo de planeo simple del Glider son dos:

- Angulo de planeo
- Velocidad

Ambos son independientes entre si a la hora de la elección. Por razones de equilibrio no todas las sendas de planeo son posibles. Todo va a depender de las características hidrodinámicas del Glider.

Dadas Kl, Kd, etc.., (coeficientes hidrodinámicos de sustentación y arrastre), podemos obtener los límites admisibles de planeo del modo en que se indica en la referencia [**Graver, 2005,** *4.1.2*]. En la pag 96 de esa referencia se determinan los ángulos límite de las trayectorias de planeo posibles en función de los coeficientes hidrodinámicos. Lo siguiente es código OCTAVE-MATLAB, de la referencia **[]** , modificada para este TFM.

```
%% SENDAS DE PLANEO ADMISIBLES entre....
% los límites solo dependen de las K hidrodinamicas
lim1 = rad2deg(atan(2*(KD/KL)*((KL0/KL) + nthroot(((KL0/KL)^2) + (KD0/KD), 2))));
lim2 = rad2deg(atan(2*(KD/KL)*((KL0/KL) - nthroot(((KL0/KL)^2) + (KD0/KD), 2))));
fprintf('Admissible values of xi_d     = (-90, %.1f) U (%.1f, 90)', lim2, lim1);
```

Si la senda pretendida queda dentro de esos límites calculados, sería un vuelo posible. En ese caso, el ángulo de ataque requerido se calcularía según la

fórmula (4.24) de la referencia **[Graver, 2005]**. Todos los cálculos de este modelo están contenidos en la misma referencia.

Angulo alfa, de ataque.

```
%% Desired angle of attack       % de Graver (4.24)

  alpha_d = (1/2)*(KL/KD)*(tan(xi_d))*...
  (-1 + nthroot(1 - 4*(KD/(KL^2))*(cot(xi_d))*(KD0*cot(xi_d) + KL0),2));
  fprintf('Desired valued of alpha_d     = %.2f degrees\n', rad2deg(alpha_d));
```

El otro parámetro (además del ángulo de planeo) que caracteriza el vuelo es la velocidad. Esa velocidad solo depende de la fuerza de flotabilidad neta. En nuestro caso, como ocurre con el SLOCUM, la cámara de lastre variable está en proa, lo que supone que un cambio de flotabilidad también altera el asiento. Si alguna de las entradas produce variación en más de una salida, decimos que hay *acoplamiento* entre entradas y salidas. Por ello hay que compensar ese cambio de trimado con el movimiento de una masa móvil. Por lo que la velocidad, para un asiento determinado, se consigue con los parámetros:

1. Lastre requerido,
2. y posición de la masa móvil

Que se calculan según las fórmulas (4.26) y (4.27) de [**Graver, 2005**]. Reproducimos el código OCTAVE-MATLAB modificado.

Lastre requerido,

```
%% Desired ballast mass      % de Graver (4.26)
 mb_d = (m - mbar - mh - ms -mw) + ...
      (1/g)*( (-sin(xi_d))*(KD0 + KD*(alpha_d^2)) + ...
      (cos(xi_d))*(KL0 + KL*(alpha_d)))*(V_d^2);
```

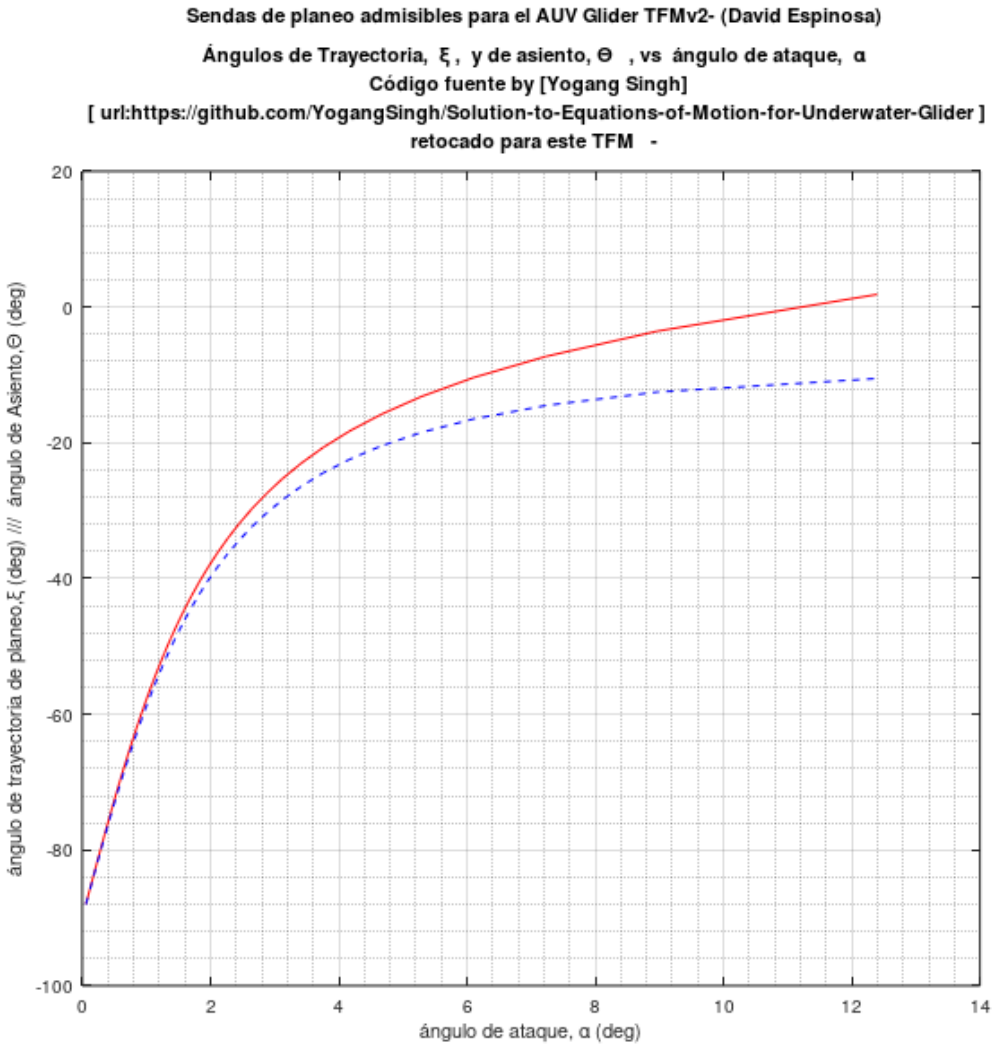
```
fprintf('Desired valued of mb_d       = %.3f kg\n', mb_d);
```

y posición de la masa móvil: rp1_d

```
%% Desired position of longitudinal moving masses  % de Graver (4.27) y pag
  % 56 y 57, y (3.5)
lambda = 1;
  theta_d = alpha_d + xi_d;
  %       de Graver (4.28) modificada o parecido
  rp1_d = (1/(mbar + lambda*ms))*(-mb_d*rb1 -tan(theta_d)* ...
        (mbar*rp3 + ms*rs3 + mb*rb3) + ...
        (1/(g*cos(theta_d)))* ...
        ((mf3 - mf1)*v1_d*v3_d + (KM0 + KM*alpha_d)*(V_d^2)));
  rs1_d = lambda*rp1_d;
    fprintf('Desired valued of rp1_d      = %.3f m\n', rp1_d);
  fprintf('Desired valued of rs1_d      = %.3f m\n', rs1_d);
```

5.4 Características de planeo de nuestro GLIDER, y comparación con otro conocido (SLOCUM)

Como resumen de esos cálculos hemos elaborado la siguiente gráfica:



Esta figura, de elaboración propia, es la que corresponde a NUESTRO Glider, y refleja las sendas de planeo posibles, y los ángulos de ataque correspondientes. Es la misma información contenida en la gráfica 5.7 de Graver, salvo que en aquel caso se refiere al SLOCUM. (también Reproducimos la gráfica del SLOCUM). La gráfica tiene simetría polar-central, por lo que nuestro gráfico es

asimilable a la porción inferior derecha de la figura de SLOCUM. (planeo en descenso). Aunque marcadas de manera diferente, continua o trazos, en ambos casos la curva más abajo de las dos es la de trayectoria, y la que está encima es la de asiento. La diferencia entre ambas es el ángulo de ataque, siempre positivo cuando el Glider desciende (Xi negativo).

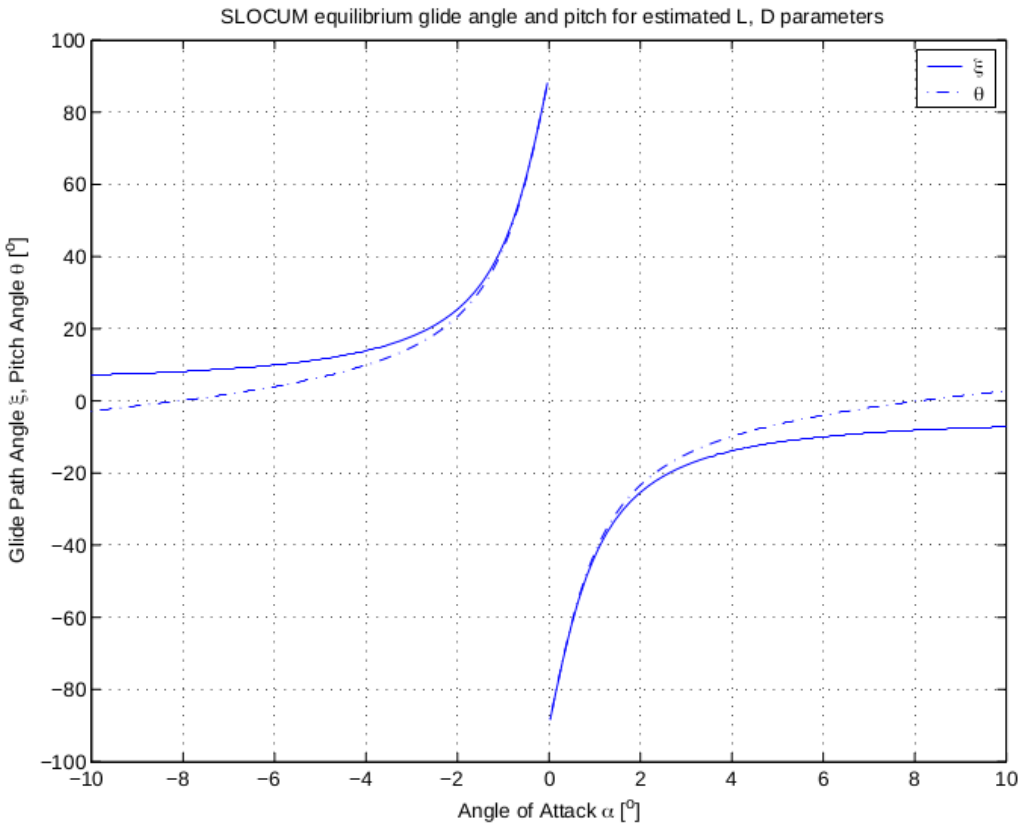


Figure 5.7: Equilibrium glides using lift and drag estimated from reference data.

Comparando las dos figuras puede verse la semejanza y las diferencias. En nuestro caso, vemos un comportamiento peor por tener un perfil alar más grueso. Por ejemplo, para alfa = 4º (positivo si el flujo entre por la cara inferior de las alas), nosotros obtenemos una senda de planeo de -23º (negativo por ser

descendente), mientras que el SLOCUM consigue -13º. Esa diferencia de rendimiento supone un mayor número de maniobras de nuestro Glider para recorrer la misma distancia horizontal. Recordemos que el mayor grosor de nuestro perfil NACA 0020 era para poder integrar unos refuerzos rigidizadores a lo largo de las alas. Hemos pagado la simplicidad de fabricación de las alas, que ahora admiten un recubrimiento no estructural, con un mayor grosor de reforzado interior.

El ángulo de ataque de 4º que hemos considerado como referencia, es un valor por encima del cual se produce inestabilidad matemática en la resolución del modelo numérico. Eso indica una relativa proximidad a una zona más horizontal del diagrama, donde está el límite de las sendas posibles (10 grados en nuestro caso, y 6º en el SLOCUM). Para conseguir que no haya inestabilidad en los cálculos tenemos que usar ángulo de planeo 20º en nuestro caso, y 15º en el caso del SLOCUM.

5.5 Acoplamiento entre las entradas y salidas, y su consecuencia

Hemos dicho que las entradas y salidas en nuestro sistema Glider estaban acopladas. En particular para una cantidad de lastre dada, el ángulo de planeo depende de la posición de la masa móvil, lo que significa que hay correlación directa entre la entrada (posición de masa móvil), y la salida (ángulo de planeo).

Sin embargo, para la otra entrada (cantidad de lastre), una modificación en su valor supone una variación de la flotabilidad (que era el principal propósito del actuador), pero además (y como efecto indeseado) una modificación en el asiento.

Para diferenciar ambas acciones, de modo que el lastrado no haga cambiar el asiento, sería necesario aplicar también una compensación en la posición del peso móvil, cuando se cambia la masa de lastre. Partimos de las siguientes relaciones:

Momento del cambio de lastre y masa móvil:

- Mom acciones = rp1. mbar + rb1. mb

Siendo:

- Mb es el lastre, ballast
- Mbar es la masa móvil (primaria).

sacamos diferenciales y establecemos la condición de que el equilibrio (Momento requerido para un determinado asiento) no cambie al actuar en el lastrado con d mb. Es decir, sacando diferenciales

- 0 = d rp1 . mbar + rb1 . d mb

El equilibrio instantáneo de las acciones coordinadas requiere:

- rp1_punto = - rb1 * mb_punto/mbar

Eso es un equilibrio instantáneo. Sería aplicable cuando estemos realizando una senda de planeo con un ángulo determinado, y deseáramos aumentar o reducir la velocidad, pero sin cambiar dicho ángulo. Realmente la acción directa sobre el lastrado aumentaría o reduciría la fuerza de flotabilidad, y con ello la velocidad, pero también afectaría al trimado, salvo que aplicásemos la corrección señalada. Las magnitudes están dadas en velocidad lineal instantánea, porque la variación de lastre está dada como masa por unidad de tiempo.

De la misma ecuación, integrando para un determinado tiempo, tendríamos otra expresión en variaciones:

cambio de posición de Mbar (masa móvil)

= incremento rp1 =

= rb1/mbar * incremento mp

o bien

$$d\,rp1 = - rb1/mbar * d\,mb$$

expresión en diferenciales, similar a una deducida previamente, donde no cuenta el tiempo.

la guardia.

Una vez terminadas las maniobras de la inversión del movimiento, no hay que hacer nada, salvo monitorizar algunos sensores de la navegación:

- Profundidad,
- Velocidad de cambio de profundidad
- Angulo de asiento
- Velocidad estimada

y esperar a que llegue (es previsible) el momento de la maniobra siguiente.

Por tanto, la mayor parte del tiempo de navegación es monitorizar el 'estado' con un guión similar al siguiente:

- Tiempo esperado para el próximo cambio.
- ¿se mantiene todo dentro de los parámetros /banda de aceptación?
- Si no, actuar en consecuencia. Aquí caben diversos modelos de gestionar los acontecimientos:
  - un sistema tradicional es con estructura tipo: *en caso de….*

    Hacer…., o if ….. then ….. else….

- O estructuras del tipo *do case case1, case2, case….. case-n, other case*
- Y también cabe otros sistemas más modernos, algo relacionado con la lógica *difusa*, pero de eso no sabemos nada.
- Usar un sistema de *sucesos*, con interrupciones asociadas a eventos, como en la *programación orientada a objetos*.
- Los sucesos previsibles serían:
  - Detección de profundidad sobrepasando alguno de los límites establecidos
  - Detección de tensión baja en la fuente de energía
  - Detección de Intensidad elevada. exceso de consumo eléctrico,
  - Detección de humedad por encima del límite marcado como 'normal'
  - detección de alguna alteración 'anómala' en la actitud o la velocidad (podría ser indicación de una colisión o un engarre)

Algunos de esos sucesos serían 'normales' y otros serían 'accidentales' o motivo de alarma. En todo caso, la principal actividad durante la navegación, aparte de las maniobras, es la vigilancia o guardia. Y cuando ocurra algo actuar según un protocolo acorde al suceso.

6 Estrategia de la navegación. Vigilancia y control.

Las misiones de los AUV Gliders se suelen establecer señalando un punto de destino, o una secuencia ordenada de ellos. Cada punto está bién definido en el espacio tridimensional. Una forma de referenciarlo es con rumbo, distancia horizontal y produndidad, relativos a otro punto previo.

El plan de navegación es la secuencia de maniobras ordenadas que habrá que realizar para alcanzar el punto de destino de la misión. Cada maniobra consiste en modificar la actitud (trimado y escora) para iniciar y mantener una senda de planeo estable, en linea recta, entre 2 puntos a distinta profunidad. Las maniobras se van alternando de modo que tras una senda descendente hay que iniciar otra ascendente y así encadenar las necesarias hasta alcanzar el destino.

La navegación de un AUV GLIDER requiere seguir el plan de navegación, controlar un conjunto de parámetros de navegación, y tomar decisiones, en cada momento, coherentes con el estado actual (en ese momento) de la navegación y las novedades que aportan las posibles variaciones de tales parámetros.

Hay una buena parte de incertidumbre respecto a ‘cuando’ y ‘cuales’ sucesos van a ocurrir. Por ello, la labor de vigilancia puede ser agotadora o consumir muchos recursos si no se trata adecuadamente.

La rutina de la navegación es una constante actitud de vigilancia y control. Monitorizar datos, detectar cambios y desviaciones, identificar la situación real en cada momento, comparar esa situación con las previstas en el plan de la

misión , tomar decisiones, ejecutar acciones y volver a empezar el ciclo de vigilancia y control.

Se podría explicar como un ciclo con las siguientes etapas:

datos – análisis – revisión de cumplimiento de la misión - decisiones – acciones - efectos …. (y vuelta a empezar en 'datos'…..),

6.1 La operativa de navegación del Glider.

En el apartado 3.1 (3.1) hicimos una breve descripción de la forma de navegar del Glider. Ahora vamos a señalar con más detenimiento la secuencia de operaciones elementales que hay que realizar.

En la navegación del GLIDER, lo habitual es combinar una sucesión de carreras alternantes, de descenso y ascenso, en las cuales el ángulo de planeo y la velocidad máxima se han decidido previamente. Normalmente las sendas alternas suelen ser simétricas respecto a un plano horizontal, de modo que el ángulo de planeo en ascenso o descenso es el mismo, y también la velocidad.

6.1.1 La maniobra de inversión

La maniobra de inversión, o cambio, de una dirección a la otra (Abajo /Arriba) se inicia en un instante característico predeterminado. Suele ser el momento en el que se traspasa una cota de profundidad 'límite', tanto hacia abajo, como hacia arriba. El límite 'profundo' se toma con margen suficiente respecto la posibilidad de chocar contra el fondo y respetando las presiones máximas admisibles en la estructura. El límite 'superficial' se marca para que el vuelo del glider no quede perturbado por alcanzar la superficie libre del líquido en el que vuela (lo que ocasionaría alteración del flujo en las alas y el cuerpo, y pérdida de sustentación que 'pararía' el vuelo).

Si adicionalmente se dispusiera dentro del Glider de una sonda capaz de medir la distancia al fondo se podría navegar con seguridad y autonomía, con independencia de la topografía del fondo del agua en el que se mueva.

En ausencia (o avería) de sensores de presión o profundidad también se podría navegar por estima, estableciendo intervalos de tiempo entre las maniobras.

La operativa normal al final de una senda de planeo, cuando se quiere invertir el sentido de vuelo, se hace del siguiente modo:

1. Lo primero es cambiar la posición del peso móvil, acción que debe realizarse relativamente rápido, y
2. simultáneamente, o con poca demora, cambiar la cantidad de masa de lastre (a la velocidad que permita su mecanismo).

Tanto la posición final del peso móvil, como la cantidad final de masa de lastre que se debe tener, son valores concretos, conocidos antes de la maniobra, y que se calculan con unas fórmulas ligadas al modelo matemático de la navegación del Glider. Esos datos forman parte del plan de navegación.

Si los actuadores son muy potentes o rápidos pueden producir sobrepasamiento y oscilaciones indeseables alrededor de la posición final objetivo de los actuadores. Si fuera necesario regular ese proceso podríamos utilizar algun control tipo PID (Regulador Proporcional-Integral-Diferencial). Si los actuadores son lentos quizá no haya oscilaciones, pues el amortiguamiento hidrodinámico también juega su papel. Sin embargo, si el actuador de variación de la masa de lastre fuese demasiado lento, podría ocurrir que la velocidad se reduzca demasiado, antes de tener 'fuerza' de flotabilidad en la dirección adecuada. En ese caso se perdería la sustentación y la impulsión correspondiente, con lo cual el glider quedaría 'a la deriva y sin gobierno'. Como ocurre con los buques de

superficie, el movimiento a lo largo del eje longitudinal de la nave es fundamental para que los órganos de gobierno (timón o alas sustentadoras) generen la sustentación requerida para poder modificar la trayectoria. Si la nave no avanza, es ingobernable.

6.1.2 La Vigilancia o Guardia de Navegación

Una vez terminada la maniobra de la inversión del movimiento (arriba/abajo), no hay que hacer nada, salvo monitorizar algunos sensores de la navegación:

- Profundidad,
- Velocidad de cambio de profundidad
- Angulo de asiento
- Velocidad estimada

y esperar a que llegue (es previsible) el momento de la maniobra siguiente.

Por tanto, la mayor parte del tiempo de navegación es monitorizar el 'estado' con un guión similar al siguiente:

- Tiempo esperado para el próximo cambio ¿?.
- ¿se mantiene todo dentro de los parámetros /banda de aceptación?
- Si no, actuar en consecuencia.

Aquí caben diversos modelos de gestionar los acontecimientos:

- un sistema tradicional es con estructura tipo: *en caso de…. Hacer….,*
- *o if ….. then ….. else….*
- O estructuras del tipo *do case case1, case2, case….. case-n, other case*

Y también cabe otros sistemas más modernos:

- Usar un sistema de *sucesos*, con interrupciones asociadas a eventos, como en la *programación orientada a objetos*.

En esta caso, los sucesos previsibles serían:

- Detección de profundidad sobrepasando alguno de los límites establecidos
- Detección de tensión baja en la fuente de energía
- Detección de Intensidad elevada. exceso de consumo eléctrico,
- Detección de humedad por encima del límite marcado como 'normal'
- detección de alguna alteración 'anómala' en la actitud o la velocidad (podría ser indicación de una colisión o un engarre)

Algunos de esos sucesos serían 'normales' y otros serían 'accidentales' o motivo de alarma. En todo caso, la principal actividad durante la navegación, aparte de las maniobras, es la vigilancia o guardia. Y cuando ocurra algo actuar según un protocolo acorde al suceso.

La rutina de vigilancia supone controlar datos y procedimientos, y llegado el caso, disparar determinados procedimientos cuando se cumplen algunas condiciones de los datos o parámetros monitorizados o vigilados.

- Los parámetros pueden ser datos, resultado de la medición de los sensores, o consignas establecidas: 'BAJAR', 'avanzar 100 m', 'no superar la profundidad de 20 m', 'obtener un asiento de 20º', etc. ' ,
- las acciones a tomar pueden ser del tipo: 'SUBIR', 'invertir el planeo', 'reducir la velocidad', 'aumentar lastre', 'medir la velocidad', 'medir profundidad', 'medir tensión de batería', 'medir intensidad consumida por el motor', etc.
- Los objetivos de la misión pueden ser del tipo: 'recorrer 100 m', 'navegar a rumbo NE 30º', 'no superar 10 m de profundidad', 'alcanzar el destino en el mínimo tiempo posible', etc.

7 Programación del comportamiento del Glider.

Intentar construir una lógica de navegación a partir de una programación tradicional imperativa, orientada a procedimientos, pero con capacidad de atender a alarmas, y tomar decisiones según los distintos estados que pueden indicar los instrumentos o sensores, daría lugar a un conjunto muy grande de bifurcaciones del tipo If … then… else … endif encadenadas una tras otra, o anidadas unas dentro de otras, etc, de muy difícil concepción y seguimiento.

En todo caso, la programación sería compleja. Además, añadir algún criterio nuevo a la misión, incluir algún otro sensor, o asumir su carencia accidental, podría generar dificultades importantes para reorganizar el código.

Creemos saber cómo se debe comportar el AUV GLIDER, pero todavía no sabemos cómo incorporar toda esa 'forma de pensar o hacer' en un código ejecutable por un procesador.

Claramente se detecta que la programación tradicional, secuencial, no es un modelo adecuado para formular las reglas de la navegación del GLIDER. Investigamos sobre la forma de enfocar problemas de este tipo, y descubrimos que existen otros modelos o estilos de programación adecuados a circunstancias concretas. Así existen los llamados **[paradigmas de programación]**, ligados a los diversos **[Lenguaje_de_programación]** (ver referencias).

7.1 Lenguajes orientados para hacer guiónes de la misiones

Las instrucciones para planificar la misión son una parte importante del desarrollo del diseño de los AUV Gliders. Hay una constante investigación en la mejora de la arquitectura del software. El siguiente paper es un ejemplo. Es una descripción del desarrollo de un leguaje orientado a definir las misiones.

> ***[A Lightweight Scripting Engine for the Slocum Glider]***. *2010. By Hans Christian Woithe, Ulrich Kremer, Department of Computer Science*

Rutgers University Piscataway, NJ 08854 Email:

{hcwoithe,uli}@cs.rutgers.edu

7.2 Paradigmas de programación

Distinto de preparar la misión es programar la 'lógica' con la que el AUV navega. Aquí es donde conviene revisar las alternativas posibles.

Un paradigma de programación es un estilo de desarrollo de programas. Es decir, un modelo para resolver problemas computacionales. Cada lenguaje de programación se puede encuadrar según lo adecuado que resulte para resolver problemas siguiendo algún paradigma concreto. Hay lenguajes que pueden encuadrarse como dentro de uno o varios paradigmas a la vez, dependiendo del tipo de órdenes que permita implementar. Es algo que tiene una relación directa con su sintaxis.

¿Cuáles son los principales paradigmas de programación?

- **Imperativo**. Los programas se componen de un conjunto de sentencias que cambian su estado. Son secuencias de comandos que ordenan acciones a la computadora.
- **Declarativo**. Opuesto al imperativo. Los programas describen los resultados esperados sin listar explícitamente los pasos a llevar a cabo para alcanzarlos.
- **Lógico**. El problema se modela con enunciados de lógica matemática.
- **Funcional**. Los programas se componen de funciones, es decir, implementaciones de comportamiento que reciben un conjunto de datos de entrada y devuelven un valor de salida.

- **Orientado a Objetos.** El comportamiento del programa es llevado a cabo por objetos, entidades que representan elementos del problema a resolver y que tienen atributos y comportamientos ante 'sucesos'.

Otros son de aparición relativamente reciente y no forman parte del grupo principal:

- **Dirigido por eventos**. El flujo del programa está determinado por sucesos externos (por ejemplo, una acción del usuario).
- **Orientado a aspectos**. Apunta a dividir el programa en módulos independientes, cada uno con un comportamiento bien definido.

Cada paradigma es ideal para la resolución de un conjunto de problemas particular, por lo que no puede decirse que uno sea necesariamente mejor que otro.

7.3 Paradigma de Programación Dirigida por Eventos. Algunos antecedentes de su uso para la navegación del AUV

Revisando literatura sobre Gliders submarinos hemos comprobado que este tipo de gestión de la navegación, 'dirigida por eventos', ya ha sido utilizado por otros.

7.3.1 Ejemplo 1

Comentaremos la referencia **[A High Accuracy Navigation System for a Tailless Underwater Glider]**

La figura adjunta, extraída de esa fuente, es el grafo que representa la **máquina de estados** simple que gobierna el sistema Glider descrito en ese paper.

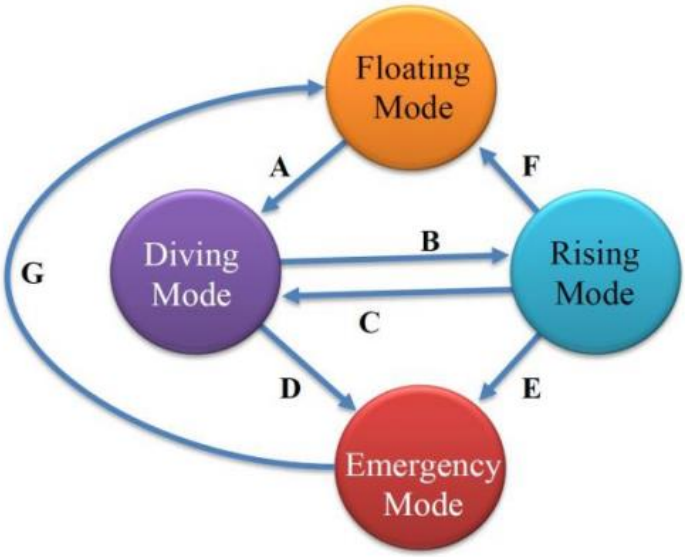


Fig. 3. Functional Modes.

(pie de foto) La **máquina de estados finitos** (FSM, Finite State Machine) representada en la figura es un modelo del sistema de navegación de un Glider.

Cada nodo representa un estado (o modo funcional, en la terminología de este ejempo). Mientras el sistema (el Glider) se encuentra en un estado, se realizan las actuaciones propias de ese estado. Como resultado de esas actuaciones, o por causas externas, pueden producirse eventos que a su vez pueden dar lugar a otras actuaciones o a cambios.

Existen algunos eventos, llamados de transición, que son causantes de que el sistema pase de un estado a otro. Esos eventos se han indicado en el grafo como flechas.

Descripción de los *estados* del sistema Glider navegando:

1. Modo *flotante*. El dron/Glider está flotando en la superficie: En primer lugar adquiere coordenadas de posición del sistema GPS y lo envía a través del sistema Iridium. Despues, espera algún barco de asistencia y activa el Wi-Fi para la descarga de nuevos comandos o actualizaciones de software.
2. Modo de buceo (*descendente*). El dron está bajando. Avanza a velocidad constante hasta que alcanza la profundidad máxima programada.
3. Modo *ascendente*. El dron está subiendo. Avanza a velocidad constante hasta que alcanza la profundidad mínima programada.
4. Modo de *emergencia*. Este modo es una especie de 'botón de pánico' del sistema. Si se detecta una actitud (posición) irrecuperable, el sistema pasa automáticamente a este modo de emergencia: la flotabilidad y el trimado se disponen para la máxima capacidad de ascenso con objeto de llegar a la superficie lo antes posible.

La transición completa de un estado a otro puede consistir en realizar un conjunto de tareas. Las condiciones (***EVENTOS***) que determinan (disparan) el inicio de la transición entre estados son las siguiente:

A - El dron **recibe la orden** de sumergirse y alcanzar la profundidad máxima programada.

B - El dron **ha alcanzado la profundidad** máxima programada.

C - El dron **ha alcanzado la profundidad mínima** programada.

D - Attitude System Control **detecta una actitud irrecuperable**.

E: Attitude System Control **detecta una actitud irrecuperable**.

F: el dron ha finalizado la misión y **recibe la orden de emerger** para esperar nuevas comunicaciones.

G - Se establece la flotabilidad al máximo y el trimado (actitud) se **ajusta para la máxima velocidad de ascenso.**

El estado de los modos funcionales y los modos de transición es gestionado por el sistema de control integrado Glider. (**GICS**)

SISTEMA DE CONTROL INTEGRADO DE GLIDER, (**GICS**)

El sistema de control integrado Glider realiza el control de actitud y navegación y el manejo y control de datos, incluidas la carga útil y las funciones de gestión comunicaciones. La arquitectura GICS (ver Fig. 4) está construida alrededor de una unidad central de procesamiento, la Unidad Central Glider (GCU), e incluye varias unidades terminales remotas que interconectan la carga útil y el equipo Glider. Las funciones realizadas por GICS son las siguientes:

- Determinación y control de trimado, escora y rumbo (actitudes).

• Control de flotabilidad y otros componentes de propulsión.

• Adquisición, formateo y codificación de datos de telemetría.

• Detección de comandos, decodificación, distribución y actuación.

• Gestión de la batería.

• Gestión de la carga útil.

Fig. 4. GICS general arrangement.

De la misma referencia, esta figura representa los vínculos entre el sistema general de navegación y el control.

7.3.2 Ejemplo 2

Otro ejemplo que avala el uso de máquinas de estado finitas, controladas por eventos, para la gestión de UAVs GLIDERs, se describe en la siguiente referencia:

[Docking for an Autonomous Ocean Sampling Network]. IEEE JOURNAL OF OCEANIC ENGINEERING, VOL. 26, NO. 4, OCTOBER 2001

En ese caso se aplican a la gestión de bases marinas para Gliders, donde se suministra 'mantenimiento', 'aprovisionamiento' y 'escala', 'instrucciones', etc. Ahí se ponen en relación los gestores de la misión, los clientes contratantes de los servicios de la misión, los suministradores de energía, y las comunicaciones asíncronas entre todos ellos y los datos compartidos, las comunicaciones submarinas con los Gliders sumergidos, etc.

7.3.3 Ejemplo 3

También tenemos constancia del **uso de FSM en el Slocum** a través de algunos papers. Aunque, lamentablemente, no tenemos mucha información de ninguno de ellos. Pese a ello, y animadoss por las coincidencias entre nuestra interpretación de como podría controlarse el AUV, y lo que vamos viendo en la referencia del ejemplo 1, vamos a intentar adaptar esa arquitectura FSM a nuestro AUV.

7.4 nuestro AUV glider: una máquina de estados finitos controlada por eventos.

El **ejemplo 1**, descrito previamente, nos ha mostrado un forma eficaz y sistemática para programar el comportamiento del AUV Glider. Se usa el concepto de máquina de estados finitos y eventos, y ofrece una visión suficiente abstracta para abordar con facilidad variaciones sobre ese ejemplo.

Empezaremos por transformar esa representación de grafo del ejemplo 1, sacada de la referencia **[A High Accuracy Navigation System for a Tailless Underwater Glider]** en una representación en **tabla de transición de estados,** que es la forma en la que nosotros vamos a trabajar con **[Automaton].**

	A-inmersion El dron recibe la orden de sumergirse y alcanzar la profundidad máxima programada.	B - El dron ha alcanzado la profundidad máxima programada.	C - El dron ha alcanzado la profundidad mínima programada.	D - Attitude System Control detecta una actitud irrecuperable.			
1-floating	b						
2-diving		c					
3-rising			b				
4-emergency				d			

A - El dron recibe la orden de sumergirse y alcanzar la profundidad máxima programada.

B - El dron ha alcanzado la profundidad máxima programada.

C - El dron ha alcanzado la profundidad mínima programada.

D - Attitude System Control detecta una actitud irrecuperable.

E: Attitude System Control detecta una actitud irrecuperable.

F: el dron ha finalizado la misión y recibe la orden de emerger para esperar nuevas comunicaciones.

G - La flotabilidad se establece al máximo y el trimado (actitud) se ajusta a la máxima velocidad de ascenso.

Algunos eventos relevantes pueden ser del tipo:

- 'la tensión de las baterías es menor que el límite inferior de seguridad'
- 'la profundidad es mayor que el límite de seguridad'
- 'el rumbo difiere del de referencia en más de 30º'

8 Entornos de desarrollo para programar sistemas controlados por eventos.

Revisando la literatura hemos encontrado algunas librerías adecuadas para trabajar con ArduinoIDE (el entorno integrado de programación -IDE- de Arduino, que también es aplicables a otras 'placas', como el ESP32) y el paradigma 'dirigido por eventos'.

8.1 [Automaton], Un entorno de trabajo para programar sistemas dirigidos por eventos. Creando code C++ compatible con ArduinoIDE.

Nos han parecido relevante **[Automaton]**, por lo atractivo que resulta para nosotros su enfoque 'novedoso', y las grandes facilidades de ampliación y desarrollo que tiene.

Creemos que con esta librería se podría implementar fácilmente un sistema de automatismos y alarmas como los de una cámara de máquinas preparada para funcionar sin dotación (humana) permanente. La conocida notación UMS que establecen las Sociedades de Clasificación. Algo similar es lo que pretendemos para nuestro AUV GLIDER

No entraremos a detallar el entorno, pero si a describir algunas de sus cualidades:

Automaton, es la palabra inglesa para referirse a un Autómata, una máquina capaz de realizar un conjunto de acciones, por si solo, sin intervención humana.

El entorno [Automaton] permite crear aplicaciones para Arduino o placas de desarrollo similares, como el ESP32, que consisten en componentes (máquinas de estados finitos) que se ejecutan simultáneamente y que interactúan entre sí. Los cambios se propagan automáticamente a través de la aplicación (autómata)

de modo similar a como los resultados se actualizan en las diversas celdas de una hoja de cálculo.
Los componentes del autómata pueden activarse entre sí y formar estructuras de control complejas.

La fortaleza del sistema [Automaton] es que **el usuario no requiere de cualidades de programador**. Solo tiene que centrarse en comprender y señalar la relación entre estados, eventos y actividades para el problema que quiere resolver.

El usuario elige los tipos de máquina finitas que requiere, entre los que ya dispone el entorno, y puede modificarlos o crear otro nuevos. A partir de ahí, cada componente se define por una tabla de transición de estados.

Las tablas de transición de estados son una forma bastante clara de describir el comportamiento de una máquina de estados.

El entorno de desarrollo de aplicaciones [Automaton] está realizado en C++ y maneja internamente el paradigma de Objetos (POO), pero en este caso los objetos son miembros de una 'Clase': máquina de estados finitos.

En la práctica, el usuario apenas se acerca a la programación en C++. Por el contrario, se centra en la definición de unas tablas que constituyen cada máquina de estados, y es el propio entorno de desarrollo el que, partiendo de esas tablas, crea el código fuente C++. Con ello, la tarea de programación queda muy asistida, y eso permite al usuario 'normal' (con poca capacidad de programación) abordar la creación del código fuente necesario. Basta con entender los procesos reales del sistema que debe emular la automatización, y establecer listas de eventos, acciones, estados, relaciones, secuencias, etc.

8.1.1 Algunas características declaradas por el autor de [Automaton]

- la Clase básica es una máquina de estado cooperativa multitarea que puede servir para construir 'componentes' diferentes a los ya disponibles en la librería que incorpora.
- Los 'componentes' son máquinas de estado basadas en tablas, usted define el comportamiento usando solo una pequeña codificación
- Clase de programación de máquina ligera
- Existen componente ya creados de tipo temporizador, y de tipo contador de estados
- Los componentes se pueden comunicar entre si a través de disparadores de eventos, conectores y llamadas a métodos directos.
- se pueden establecer Estados de suspensión para ahorrar ciclos de microcontroladores
- existe modos de diagnóstico o Seguimiento de depuración (monitor de estado) que le permite ver qué están haciendo los componentes
- Fomenta el diseño modular y la separación de preocupaciones.
- Los componentes nuevos que puedan crearse se pueden compartir como bibliotecas Arduino independientes (solo dependen de la biblioteca Automaton)

Hay que destacar que el sistema logra una operatividad multitarea, concurrente, simultánea y entrelazada de todos los componentes, lo cual parecería imposible en un sistema sencillo como el Arduino.

La ventaja de usar máquinas de estado es que se puede ejecutar docenas de máquinas de estado al mismo tiempo, cada una realizando su propia subtarea. El objeto del planificador de Automaton los mantiene a todos corriendo en su prioridad seleccionada. Las máquinas pueden comunicarse de forma asíncrona a través de la cola de mensajes y así se pueden realizar fácilmente tareas complicadas mientras (a la vez que) el Arduino responde a la entrada del usuario (eventos de usuario). Muchas de las tecnologías más populares de la actualidad, como Node.JS y Twisted de Python, se basan en marcos basados en eventos. Ahora el Arduino también puede tener uno.

8.1.2 Adaptacion necesaria para el uso de [Automaton] con el ESP32.

La librería de [Automaton], fué desarrollada y publicada en 2018 para el procesador ESP8266, que es anterior al ESP32, y no es compatible con el .

El autor de la librería no ha actualizado esas líneas pese a las peticiones de algunos usuarios:

https://github.com/tinkerspy/Automaton/issues/77

https://github.com/tinkerspy/Automaton/issues/72

https://github.com/tinkerspy/Automaton/issues/41

Sin embargo, el code fuente si puede ser reutilizado, si se usan las librerías adecuadas. En particular esto se requiere en las librerías relacionadas con la gestion del Wifi.

En la web…

https://www.diarioelectronicohoy.com/blog/introduccion-a-la-libreria-wifimanager

nos dan un ejemplo de como usar las librerías adecuadas según se desee que el código fuente compilado funcione en uno u otro. según se use el ESP8266 o el ESP32

```
#if defined(ESP8266)
#include < ESP8266WiFi.h > //ESP8266WiFi.h .- ESP8266 Core WiFi Library
#else
#include < WiFi.h >  //WiFi.h .- ESP32 Core WiFi Library
#endif

#if defined(ESP8266)

#include < ESP8266WebServer.h > //ESP8266WebServer.h .- Servidor web local utilizado para servir el portal de configuración
#else

 #include < WebServer.h > //WebServer.h .- Servidor DNS local utilizado para redireccionar todas las solicitudes al portal de configuración (https://github.com/zhouhan0126/DNSServer---esp32)

#endif

 #include < DNSServer.h >//DNSServer.h .- Local WebServer usado para servir el portal de configuración (https://github.com/zhouhan0126/DNSServer---esp32)

#include < WiFiManager.h > //WiFiManager.h .- WiFi Configuration Magic (https://github.com/zhouhan0126/DNSServer---esp32) >> https://github.com/zhouhan0126/DNSServer---esp32 (ORIGINAL)
```

Para poder usar Automaton con el ESP32 tendríamos que usar estas estructuras de cambio de librerías, y probar el ejemplo de la web de Automaton.

Otra posibilidad que tendríamos sería usar el ESP8266, pero no es la opción deseable para nuestro AUV, ya que es una placa de menos potencia de proceso.

Ya sea con unas herramientas o con otras, entendemos que la mejor arquitectura a nuestro alcance está basada en la forma de Maquinas de estados finitos. Aunque es posible que implementar totalmente el código funcional supere la dedicación que podemos ceder a esa tarea dentro de este TFM.

8.2 ESP__EVENT, la librería para programación orientada a eventos de ESP32: .

El fabricante del chip ESP32 es ESPRESSIF. Como el producto ESP32 está orientado a su uso en el Intenet de las cosas (IOT), ellos han desarrollado y publicado un conjunto de herramientas que facilitan esa labor: el ESP-IDF.

https://docs.espressif.com/projects/esp-idf/

Es un entorno de desarrollo integrado (IDF, Integrated development Framework) para crear aplicaciones alrededor del producto de ESPRESIFF, el chip ESP32, entre otros.

Aunque puede ser usado para crear aplicaciones sin necesidad de ningun otro recurso externo, también pueden usarse sus librerías para crear aplicaciones en otros entornos de desarrollo, como el arduino IDE.

En particular, han creado una librería específica para el ESP32 para ser usada en la programación orientada a eventos:

https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/esp_event.html

Se trata de una **API** (en inglés, *application programming interface,* interfaz de programación de aplicaciones). Una API es un conjunto de funciones y procedimientos que aporta una librería o biblioteca para que pueda ser utilizada con facilidad por otro software.

Esta API nos facilita implementar la gestion de eventos en la programación del ESP32.

Finalmente, este será el entorno más probable en el que desarrollaremos el software para la navegación del UAV sobre ESP32.

Pero en cualquiere caso, además de una librería de gestion de eventos, también usaremos el concepto de máquinas de estados finitos. Ambos temas se complementan en la arquitectura de la 'lógica' que buscamos para la navegación.

8.3 Las tablas de transición de estados

Aparte de los grafos de nodos y flechas, que hemos visto anteriormente, las tablas de transición de estados son una buena forma de representar el funcionamiento de una máquina de estados finitos. Es la representación que que vamos a usar. Se trata de una tabla con filas y columnas.

A lo largo de la dimensión vertical distribuimos los Estados posibles (una fila para cada estado). A lo largo la dimensión horizontal indicamos los posibles eventos (una columna por cada uno, señalado en la cabecera de cada columna), y las celdas (intersecciones fila/columna) de la tabla contienen el siguiente estado que corresponde antes el estado de esa fila si ocurre el evento señalado en la cabecera de columna, y también la acción (si fuera el caso) enlazada (Ax/) que debe ejecutarse en esta transición de estados.

Tabla de Transición de Estados

Event State	E_1	E_2	...	E_n
S_1	-	A_y/S_j	...	-
S_2	-	-	...	A_x/S_i

...	...	...	...	...
Sm	Az/Sk	-	...	-

(S: estado, E: evento, A: acción, -: transición no inadmisible)

En este ejemplo, si la máquina está en el estado S1 y se produce el evento E1, no pasa nada, pero si ocurre el evento E2, la máquina desencadena la acción Ay, y pasa al estado Sj

En el tutorial de Automaton se muestra cómo construir una máquina de estados dentro del marco.

Dado un sistema general a resolver, la idea es subdividirlo en subtareas, cada una de las cuales suele tener una o más entradas o salidas. Para cada una de esas subtareas se debe crear una máquina de estados finitos (o autómata). Las máquinas pueden interactuar entre si, posiblemente desencadenando nuevos eventos. Esto da como resultado un código modular y fácil de entender que se ejecuta de modo muy estable.

Así no se requiere programar en C++, sino que se describirá como una tabla de estados que utiliza pines, temporizadores y contadores como entrada.

Empecemos, pues, a convertir la secuencia de la navegación en esas operaciones ...

maquina MISION

8.4 La tabla de transiciones de estados para nuestro UAV

Inspirándonos en el **(Ejemplo 1)**, vamos a intentar diseñar nuestras máquinas (una o varias) de estados para gobernar la navegación del Glider

Empezaremos por transformar esa representación de grafo del ejemplo 1, sacada de la referencia **[A High Accuracy Navigation System for a Tailless Underwater Glider]** en una representación en **tabla de transición de estados**

En la primera ***columna*** se representa en cada celda un ESTADO del sistema.

En la primera ***fila***, cada celda encabeza una columna. El contenido de esa celda es un EVENTO que dispara el inicio de la transición de estado.

La ***intersección*** de filas y columnas son celdas en las que se recogen la información de las consecuencias que el evento (columna) produce en el estado (fila). Esa información es el ***estado destino*** (hacia el que se mueve el sistema), y un conjunto de ***actuaciones*** (entre ellas la propia transición)

Siguiendo el modelo del ejemplo1 hemos establecido:

EVENTOS

A - El dron recibe la orden de sumergirse y alcanzar la profundidad máxima programada.

B - El dron ha alcanzado la profundidad máxima programada.

C - El dron ha alcanzado la profundidad mínima programada.

D - Attitude System Control detecta una actitud irrecuperable.

E: Attitude System Control detecta una actitud irrecuperable.

F: el dron ha finalizado la misión y recibe la orden de emerger para esperar nuevas comunicaciones.

G - La flotabilidad se establece al máximo y el trimado (actitud) se ajusta a la máxima velocidad de ascenso.

Algunos eventos adicionales posibles (similares a D, E), pueden ser del tipo:

- 'la tensión de las baterías es menor que el límite inferior de seguridad'
- 'la profundidad es mayor que el límite de seguridad'
- 'el rumbo difiere del de referencia en más de 30º'

También se pueden considerar algunos estados adicionales. En particular será relevante un estado de PRE-BOTADURA, y otro siguiente de BOTADURA (o puesta en el agua), que debería ser el anterior al A.

. Inicialización.

en el estado de pro

Se parte de el estado AUV colocado en el agua, y se realiza un conjunto de actuaciones hasta llegar al estado siguiente, que sería AUV flotando en superficie y listo para recibir orden de inicio de misión. Ese estado sería el A.

El resultado de esas ampliaciones es la siguiente tabla ((se presenta en otro documento, en formato apaisado y letra pequeña):

SYC tabla estados AUV automaton1.docx

9 Agenda - Preparar el seudo code del programa de control para Arduino.

- Pero sobre todo, seudo code para todo. para simular los estados de autómatas o maquinas finitas - AUTOMATON

- Preparar 2 niveles de ejecución del software: comunicaciones al exterior via wifi o bluetoot con un terminal estandar exterior tipo smarphone Android, y comunicaciones con el sistema, comandado por Arduino. con AUTOMATON podemos hacer que todo corra en ESP32, con más ram y más de todo, y que puede ejecutar code Arduino, y además tiene la WIFI y el BT incorporados, en el chip, y en el code de AUTOMATON para ESP…
- En esas condiciones parece recomendable prescindir de Arduino, y decantarnos por el ESP 32, sabiendo que podemos programarlo con el mismo code C++, en la misma IDE de Arduino.
- Para la gestión de comunicaciones se usará el ESP32 , programado en micropyton, que admite una plataforma muy fácil de control como servidor wifi de páginas web, donde se puede intercambiar info entre el arduino y el usuario. Bajo consumo, barato y eficaz energéticamente y de fácil programación, partiendo de los ejemplos y soluciones ya existentes.
- Para las órdenes de los servos y motores y sensores usar arduino.
- Se debe primar la no acción siempre que se pueda, tanto en movimiento de mecanismos. Como en lectura de sensores, como en tiempo de proceso, como en ocupación de la cpu. Si fuera posible debería actuar como un cazador de sucesos al acecho, siendo el acecho una situación de mínimo gasto de energía, a la espera. Para ello es admisible estrategias diversas: bucles con paso variable en tiempo de espera antes de reiniciar el bucle, y ligar ese tiempo a las previsiones o estimaciones, y por otro lado, sacar ventaja de los procesos que funcionen mejor con paso constante de acciones cíclicas. Equilibrar ambos paradigmas.

- Aclarar que el controlador PID siempre será necesario, cuando tras una acción de control no se reciba (ya sea por desgaste, perturbaciones exteriores, etc) la lectura del sensor correspondiente a lo deseado, y por tanto haya que modificar la señal de control hasta que se produzca. Siempre será requerido.

10 El control: la maniobra de inversión bien afinada, y uso de un controlador tipo PID

10.1 la maniobra de inversión del planeo 'bien afinada'. Eficacia y limitaciones en las variables eléctricas y cinemáticas de los actuadores.

Es de suponer que la maniobra más armoniosa posible en el cambio de rasante al final de la senda de planeo (de ascendente a descendente o viceversa), será aquella que produzca la mínima pérdida de velocidad, y no produzca oscilaciones residuales de trimado cuando se establece la senda de salida. En esto puede tener importancia una adecuada velocidad y sincronía de los actuadores involucrados en el movimiento de las masas de trimado y lastre.

…….

La velocidad instantánea que cada actuador induce sobre la masa que controla dependerá de las especificaciones del servo motor, o motor DC, o dispositivo similar:

- aceleración máxima

- Velocidad máxima
- Recorrido máximo

que a su vez dependerán del hardware, y de las características eléctricas aplicadas: tensión e intensidad.

El peso móvil describe una trayectoria cinemática definida por

- posición, rp1
- velocidad, rp1_punto,
- aceleración, u_acel_masa_movil, rp1_pp, con valores 0, ± 0.100 mm/s2

La operación se inicia partiendo de velocidad nula. Aplicando una aceleración constante, la velocidad crece hasta alcanzar la velocidad máxima. Esa velocidad se mantiene unos instantes y, en el último tramo de aproximación, la velocidad deberá reducirse hasta cero con una desaceleración constante, igualmente controlada.

Puede ocurrir que la posición se alcance sin que se haya llegado a desarrollar la velocidad máxima posible para el actuador.

Todos los movimientos de los actuadores están delimitados por los fines de carrera, de tipo hardware, que evitan que puedan producirse daños en los mecanismos.

La gráfica de las velocidades en función de la posición del actuador será parecida a la siguiente.

En este caso el desplazamiento total se ha dividido en 3 tramos de tiempo iguales: T1, T2 y T3, en los cuales se producen movimientos de tipo

aceleración constante, velocidad constante, desaceleración constante.

Durante esos movimientos, los espacios recorridos por el actuador son, respectivamente, S1, S2 y S3. Llamaremos ST al espacio total

$S1 = s1\text{-}s0 = \frac{1}{2} \cdot a \cdot (t\_1 - t\_0)^2 = \frac{1}{2} a\ T1^2$

$S3 = s3\text{-}s2 = \frac{1}{2} \cdot a \cdot (t\_3 - t\_2)^2 = \frac{1}{2} a\ T3^2$

$S2 = a \cdot T1 \cdot (t2\text{-}t1) = a\ T1\ T2$

$ST = S1 + S2 + S3 =$

$= 2S1 + S2 =$

$= a\ T1^2 + a\ T1\ T2$

$= a\ T1\ (T1 + T2)$

$= a\ T1^2\ (1 + T2/T1)$

llamaremos al tiempo total TT

$$TT = T1 + T2 + T3$$

$$= 2\,T1 + T2$$

$$= 2\,T1\,(1 + 0.5\,T2/T1)$$

Asumimos que las rampas de aceleración y desaceleración son iguales, con lo que T1 = T3,

también podemos asumir que existe una aceleración máxima posible, para una intensidad máxima admisible, y que la velocidad máxima también existe, limitada por una tensión máxima admisible.

Así las relaciones mecánicas y eléctricas están limitadas, y eso determina el tiempo mínimo necesario para la maniobra de in desplazamiento total ST, que puede tener o no parte de velocidad constante, siempre que dé tiempo a alcanzar la velocidad máxima durante el recorrido.

Las limitaciones en aceleración y velocidad mecánicas tienen sus correspondencia eléctrica con intensidades máximas y tensiones máximas admisibles. En la práctica esos son los factores que limitan la velocidad y potencia de los actuadores.

límites:

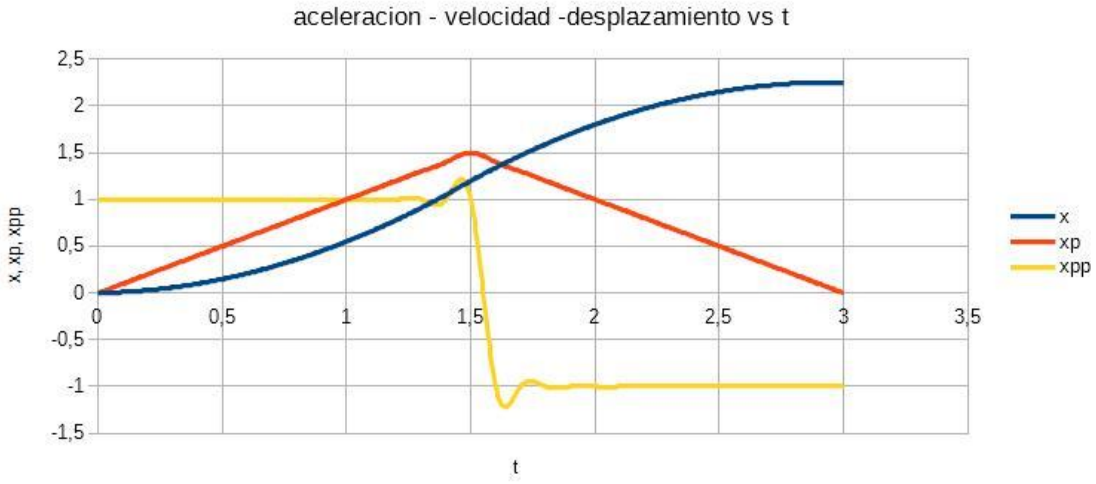
aceleracion_max

velocidad_max = aceleracion_maxima . T1

un recorrido total del actuador, ST, tarda en realizarse un tiempo total TT.

El tiempo mínimo para hacer un recorrido supone agotar los máximos de aceleración y velocidad, lo que nos llevaría a no tener tramo de velocidad constante , es decir T2 =0

ese caso correspondería al siguiente diagrama de velocidades, aceleraciones, desplazamientos y tiempos



en este caso se aprecia que en el mismo tiempo de 3 segundos hemos logrado un recorrido de 2.2, mayor que en el otro caso (que era 2.0). Eso ha sido posible al poder superar, durante el intervalo entre los instante t= 1 y t=2, la velocidad que teníamos limitada a v=1 en el otro caso

la velocidad promedio en un recorrido ST a la máxima aceleración y misma máxima desaceleración sería

$ST / TT = a\,T1^2 / 2\,T1$

$velocidad\_media\_max = a\ T1\ /2$

un recorrido determinado ST se hará en un tiempo 2T1, con una aceleración a, tal que a < aceleracion_maxima.

y además sin superar la máxima velocidad posible del actuador,

a . T1 < velocidad_máxima

si tenemos limitaciones en la aceleración o la velocidad máximas, corresponderá usar una trayectoria combinada de 2 rampas y un tramo de aceleración nula (a la máxima velocidad)..

T1 mínimo = velocidad_maxima / aceleracion_maxima

En las librerías de código programado que usaremos para controlar los motores y actuadores, ya se contemplan estas características de aceleraciones o velocidades limitadas, y las rampas correspondientes de actuación.

Por último, no necesitamos representar un caso en el que no se alcanza la velocidad límite, ya que estaría ya contemplado en este último diagrama. En tal caso el pico de la velocidad estaría por debajo de la velocidad límite, y eso significaría que con las mismas aceleraciones, recorrer un espacio menor llevaría menos tiempo, y no se podrían llegar a desarrollar la máxima velocidad admisible en ningun momento del recorrido.

Todo esto son casos donde se buscaba hacer el movimiento en el mínimo tiempo posible, y para ello apurábamos los límites de intensidad y tensión de los actuadores, (aceleración y velocidad, en términos del movimiento).

Cuando el tiempo mínimo para realizar una maniobra no es un requisito, no necesitamos apurar tales variables eléctricas o mecánicas.

Veamos ahora lo que corresponde hacer con el segundo actuador.

Simultáneamente al movimiento de la masa móvil, también se dá la orden de cambiar el estado de lastre. El **actuador de lastrado** es un motor DC (corriente continua), y la consigna es trasegar una cantidad de lastre determinada, para lo que se controla el tiempo y la velocidad de giro del motor de la bomba. La velocidad de régimen se alcanza enseguida, y el parámetro característico del actuador es

- u_ratio_de_lastrado, con valores 0, ± 0.002 kg/s

El actuador trabaja hasta que se detectan los finales de carrera de software de objetivo alcanzado para ese control: Si la masa de lastre objetivo se ha alcanzado, establece ratio lastrado =0, y mb =mb_deseada

Las 2 operaciones, movimiento de la masa móvil, o cambio de lastre, se realizan hasta terminarse, de modo independiente.

10.2 uso de un controlador tipo PID

En la referencia **[Graver , 2005]**,(imprescindible y citada en toda la literatura de Gliders que hemos revisado), hemos encontrado mucha información de gran calidad y valor. Allí desarrollan un modelo matemático matricial de EDOs que permite manejar el Glider como un sistema controlable. También hemos constatado en aquel estudio que los controladores usados en esos AUV Glider son de tipo **PID** (Proporcional-Integral-Diferencial).

El controlador PID es una de las soluciones más frecuentes en los sistemas dinámicos industriales. Son eficaces y relativamente simples, y pueden controlar tanto sistemas lineales como no lineales. Además pueden aplicarse en sistemas de los que no se tiene un modelo matemático.

La primera intención, tentadora, es plantearnos que no tenemos un modelo matemático, y explorar la metodología a seguir. En caso de no tener modelo, la forma de proceder sería crear una colección de datos tipo entrada-salida ante una entrada sencilla, tipo escalón (por ejemplo), e intentar identificar ese sistema a partir de esa colección de datos. En ***MATLAB*** hay una aplicación que dirige todo ese proceso: la aplicación ***PID Tuner*** del ***Control System Toolbox***, de MATLAB. (Toda la metodología aquí mencionada se puede ver en la referencia [MATLAB . PID tuner])

Para avanzar en el trabajo de identificar un sistema desconocido, un primer requisito es disponer del sistema físico real, y/o de colecciones de datos útiles representativos de sus series temporales entrada-salida.

A partir de la existencia de series reales de datos, puede empezarse un trabajo iterativo de identificación entre ensayos reales y ajustes físicos y matemáticos sobre el modelo virtual identificado.

El trabajo de ‘campo’ suele ser costoso en tiempo y rendimiento. En ocasiones prohibitivo. Los ajustes directos sobre elementos o parámetros de la planta en funcionamiento solo tienen viabilidad cuando el sistema es muy conocido para un personal muy experto en su manejo y ajuste.

Tenemos que entender que no solo necesitamos averiguar cómo responde el Glider a las acciones de control, sino además una guía sobre que modificar en el Glider y sus sistemas de control para llegar a un Glider real que cumpla un estándar de funcionalidad y manejabilidad satisfactorio.

Todo ello es un trabajo lento y delicado, sobre todo cuando se hacen operaciones directamente sobre la planta real.

Por eso, siempre que se puede, se recurre a modelos matemáticos del sistema para hacer los ajustes con simulaciones en ordenador. Eso reduce riesgos de accidentes, costes, y asegura poder aplicar sistemáticamente metodologías eficaces de ajuste.

10.3 Simulación de la aplicación de un controlado PID al modelo.

Como anunciábamos en el apartado

2 Posibilidades de control; 2.1 El control en un sistema genérico, y el caso particular del AUV GLIDER

nuestro sistema 'se controla solo', tal como hemos visto en la simulación. Es suficientemente amortiguado y estable como para no crear dificultades en conseguir una navegación 'suave', a voluntad.

Pero esto lo hemos averiguado gracias al modelo matemático. En otro caso diferente pudiera haber ocurrido que la simulación pusiera de relieve que el vehículo fuese 'difícil de gestionar'.

Ahí es donde se ven las ventajas de tener un modelo matemático 'creible' . En caso de necesitar un estudio importante para el control, usaríamos el entorno Xcos (vinculado a SCILAB), que es un equivalente OpenSource a Simulink (vinculado a MATLAB).

En este caso, el modo más adecuado que hemos considerado para representar nuestro sistema es en la forma de espacio de estados, caracterizado por las matrices ABCD descritas en ***[Graver, 2005]****. En el programa de simulación Xcos existen bloque específicos para esa forma, donde lo único que debemos indicar son las matrices ABCD.*

Usaremos una excitación como entrada en forma de onda cuadrada, de periodo y amplitud elegidos para que la escora esté acotada, y el Glider oscile 3 o 4 periodos antes de que la excitación cambie.

tantearemos el efecto de los parámetros del PID en la salida a partir de una entrada realimentada

10.4 Ajuste de parámetros del PID.

Usaremos la simulación sobre el modelo del sistema para ajustar las ganancias del controlador PID. En la literatura técnica usual en control de sistemas este proceso de ajuste o 'sintonía' del PID está muy tratado. Uno de los métodos más conocidos es el de Ziegler y Nichols.

métodos,

manual

automático

10.5 Un PID con las librerias de Arduino IDE, en C++,

existen librerias bien conocidas que contienen las plantillas adecuadas para establecer controladores PID.

11 Agenda:

Antes de todo hay que presentar el **modelo matemático**, como una parte necesaria para analizar el comportamiento del Glider, que ahora está en las páginas 12 a 53

Y de la pág 54-57 se tratan algunas formas de hacer la i**dentificación de los parámet**ros del modelo matemático del Glider, a partir de ensayos con el modelo real, y ahí se describe la e**xperiencia de estabilidad 'clásica'** como una forma de hacer gran parte de la identificación de los parámetros.

Progr**amación dirigida por eventos**. parte de esta materia está en las pág 52-54 anteriores., y también en las Páginas 68 a 81 tratan sobre la programación dirigida por eventos, aplicable a este caso del Glider.

Se mencionan las máquinas de estados finitos, como modelo aplicable, y tenemos un marco de trabajo AUTOMATON que parece hacer fácil esa tarea de construir las máquinas de estados en C++ a partir de las especificaciones del funcionamiento y operación del Glider.

Seguidamente se metería la nueva posibilidad de usar una máquina más moderna y potente que el Arduino, pero incluso más barata y pequeña, un sucesor aventajado, el NodeMCU-32 S, describiendo algunas de sus características, y su wifi y BT integrados, así como la posibilidad de ser programado con el de programación de Arduino, en C++, y usando librerías compatibles con aquél.

Sobre ese desarrollo se comenta la posibilidad de dirigir desde un navegador web en un smartphone , dando las consignas y recibiendo las respuestas del Glider, sobre el estados de sus máquinas y misiones.

Hemos hecho pruebas con el IDE 1.8.12 y el ejemplo del Access Point Wifi del >esP32, donde he logrado encender y apagar el led de prueba. Y para ello se ha tenido que seleccionar la librería WIFI adecuada, que no es cualquiera. Afortunadamente el compilador de esta versión 1.8.12 es capaz de buscar y indagar sobre la librería más conveniente entre las que puede localizar, y lo deja escrito.

Continuaremos trabajando con la especificación de las máquinas de estados, y después con el code c++ para completar algunos de los métodos…

Habrá que hacer las pruebas de las librerías que mueven el motor con encoder, y disponer unos finales de carrera de hardware.

Y el tema de**l PID (** ahora entre las pag 55 y stes… y y 62 y 63 , se dejará para el final y tendrá importancia mínima y se deja para el final, para ajustar con precisión cada una de las dos estradas Xp, posición de la masa móvil, y Mb, masa de lastre, de manera precisa. Para la fase final, cuando se estén haciendo pruebas reales de navegación con ajustes finos, o con el simulador

OTRAS ACTIVIDADES DE AGENDA:

1. Crear la jaula del cuerpo de popa capaz de rellenar el hueco existente de unos 20 cm, con posibilidad de modificar su longitud con las varillas roscadas, y que los discos apoyen dos a dos, cada uno entrando en sus 3 agujeros para varillas, desfasados respecto a l otro grupo de 3 varillas, desfasados formando hexágono.
2. Hacer 1+1 anillos de 3 agujeros para interface entre 2 cuerpos alargables
3. Preparar una conexión permanente a través de la tapa de popa, capaz de cargar las baterías desde el exterior, sin que se produzca descarga por humedad, incluyendo un sistema de 2 diodos interiores que impidan el retroceso de corriente desde el interior, aunque los bornes de fuera estén en corto.

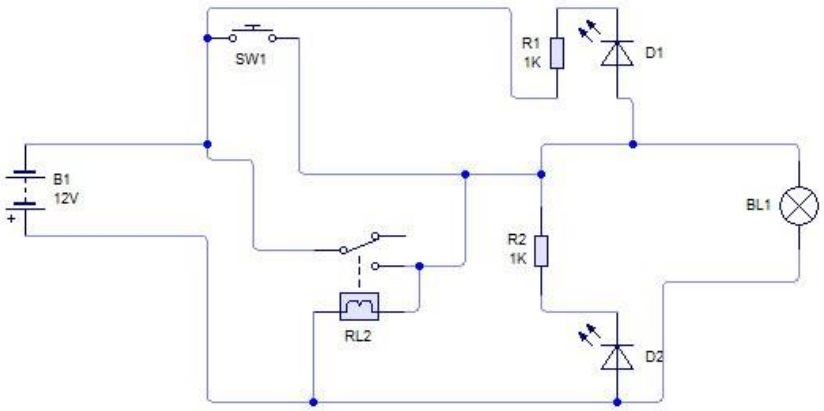
Basta (interior) ----- (+) --- [< ------- /// (+) ---------- (exterior)

----- (-) --- ->[------- /// (-) ----------

Así la corriente solo entra si la tensión fuera es mayor que la interior, y la interior no sale, aunque la resistencia externa sea pequeña, pues los diodos lo impiden en esa dirección.

y contra cortocircuito.

https://www.youtube.com/watch?v=rUHjKf45EFg



Eso me permitirá dejar un conector externo preparadas para carga la batería del Glider sin abrir el interior.

4 Preparar en 3d printer y freecad un anillo revolver para meter barras de lastre de a inox en la proa, alrededor del cilindro de metacrilato, para

ajuste fino del lastre. O trasladar a un molde de plomo para fundir en yeso o lo que sea, anillo de macho. Densidad del plomo 11.0 gr/cc. Para obtener un anillo de 1 kg, se puede hace con 30x50x80 mm, o con 25x50x85 mm, ambos dan unos 1010 gr reales

5 Preparar en 3d printer + freecad un contactor a medida de la forma de la batería del roomba, con 2 contactos+2 que no se oxiden, bajo corriente: tipo de metal ¿¿?? Niquelado, pero no cobre oxidable, ni aluminio oxidable. Tampoco acero ixox, mal conductor.

6 Preparar una rutina que muestre en la pantalla OLED los datos relevantes que le mande el esp32, Y A LA VEZ LO REFLEJE EN LA PAG WEB DE SERVIDOR. RETOCAR EL CODE DE apserver WIFI PARA ESP32 QUE HE PROBADO.

7 PREPARAR UNA RUTINA QUE LEA SENSORES Y DE SALIDA A LOS DATOS a la oled y a la webserver

8 PREPARAR UNA RUTINA QUE MUEVA SERVOS A LAS POSICIONES MANDADAS

9 Preparar uN REPETIDOR QUE INFORME DE LA TENSION E INTENSIDAD INSTANTÁNEA, Y DE LA ACUMULACIÓN

10 PREPARAR UNA RUTINA EN C++ PARA MOTOR CON ENCODER MOVIDO A UNA DETERMINADA POSICIÓN.

11 Preparar una funcionalidad para el finde carrera, stopend, para el émbolo de proa, y el carrode escora.

12

13

12 Bibiografía

- ***[Graver , 2005]*** *[**Underwater Glider dynamics and control** . 2005, J. G. **Graver**]* Tesis doctoral . se descubre que esta fuente es la más completa y detallada en el desarrollo matemático del modelo, pues es la fuente de muchas otras.

- **[Singh 2018]** . Un código MATLAB, de libre accceso, que implementa el modelo matemático descrito en **[Graver , 2005]**

  La disponibilidad de este código público nos ha permitido hacer simulaciones reales en nuestro propio modelo.

- 1- ***Ogata- Ingeniería de control moderna***, Quinta edición, ed PEARSON, 2010

- ***Model simplification for AUV pitch-axis control design***. Jan Petrich Daniel J. Stilwell . The Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061 fjpetrich,. viene a decir que un ***controlador de orden 2 cumple bien para ese trabajo***, en lugar de uno de orden 3 que sería supuestamente el perfecto. La diferencia no merece la pena en la complicación que supone frente a las mejoras que aporta. Es una conclusión muy valorable.

- ROGUE descrito en: **Model-Based Feedback Control of Autonomous Underwater Gliders**, Naomi Ehrich Leonard, Joshua G. Graver IEEE JOURNAL OF OCEANIC ENGINEERING, VOL. 26, NO. 4, OCTOBER 2001 633 -

- ***[Optimal control of an underwater Glider vehicle]*** . Renan da70 Silva Tchilian, Elvira Rafikova, Salimzhan A. Gafurov, Marat Rafikov, 2017

  Arranca de la referencia **[Graver , 2005]**, que cita. Aunque el objeto del paper es mostrar que, desde ese modelo, se puede obtener, con al técnica LQR, un controlador óptimo, la realidad es que no muestra esos pasos detallados. Pero presenta los resultados de algunas simulaciones acertadas, y los parámetros calculados en varios ejemplos para 2 Gliders reales conocidos, con los PID óptimos desarrollados para ellos.

- **[MATLAB PID tuner]**

- **[ODE – simulation with MATLAB, OCTAVE and SCILAB]** Simulation of ODE/PDE Models with MATLAB#, OCTAVE and SCILAB. Philippe Saucez, Alain Vande Wouwer, Carlos Vilas, Ed Springer, Scientific and Engineering Applications

- **[Lenguaje_de_programación]**

- **[paradigmas de programación]** (una breve reseñas de algunos tipos, ref

- **[A High Accuracy Navigation System for a Tailless Underwater Glider]** . Enrico Petritoli, Fabio Leccese . IMEKO International Conference on Metrology for The Sea Naples, Italy, October 11-13, 2017 . una descripción de navegación controlada por eventos

- **[Docking for an Autonomous Ocean Sampling Network]. IEEE JOURNAL OF OCEANIC ENGINEERING, VOL. 26, NO. 4, OCTOBER 2001** descripción de navegación controlada por eventos

- **[Automaton]** Un entorno de programación basado en eventos que permite crear aplicaciones Arduino que consisten en la ejecución simultánea de máquinas de estado que interactúan entre sí. Use las máquinas incluidas o cree las suyas propias siguiendo el tutorial. Contiene máquinas agrupadas reutilizables para manejar leds (atenuación y parpadeo), botones, comandos, seriales, entrada analógica (con promedio móvil), pulsos y temporizadores , manejo de servomotores, y más, etc.. Incluye unas utilidades para no tener que crear código C++, sino que basta con establecer la tabla de estados, acciones y eventos de cada máquina de estados.

  https://www.arduinolibraries.info/libraries/automaton

  y un ejemplo de aplicación en español de la librería a la gestion de una red de semáforos de tráfico urbano

  PROGRAMACIÓN CONCURRENTE ORIENTADA A EVENTOS EN ARDUINO PARA EL INTERNET DE LAS COSAS . AITOR CANTERO ROMERO. SEPTIEMBRE 2017. PROYECTO DE FIN DE CARRERA . ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN. UNIVERSIDAD POLITÉCNICA DE MADRID .

  http://oa.upm.es/49224/

  http://oa.upm.es/49224/1/PFC_AITOR_CANTERO_ROMERO.pdf

-

- **[Tabla de transición de estados]**

- **[Maquina de estados finitos]**

- **[ESP 32 motor lib]s** librerias de control de motres con el ESP32

- **DC motor closed-loop control software- How to build a servomotor with an arduino brain. Based on servostrap**

  https://github.com/danithebest91/Raposera

  Como convertir un dc motor con encoder en un servo eficaz.

  voy a intentar esta

  misan/dcservo

que funciona combinada con esta de un PID reconocido como bueno

br3ttb/Arduino-PID-Library

o este, que es una versión del mismo mejorada

- **[Safety of TOURIST SUMERGIBLES]**

Safety of Tourist Submersibles. Committee on Assessing Passenger Submersible Safety Marine Board. Commission on Engineering and Technical Systems. **National Research Council. NATIONAL ACADEMY PRESS.** Washington, D.C. 1990

https://www.nap.edu/catalog/1744/safety-of-tourist-submersibles [PDF]

https://www.nap.edu/download/1744

- **[SECTION 18 Autonomous Underwater Vehicles (AUV) (2014)]**

ABS RULES FOR BUILDING AND CLASSING **UNDERWATER VEHICLES**, SYSTEMS AND HYPERBARIC. FACILITIES . 1 ene. 2019

ww2.eagle.org › 7-underwatervehicles › uwv-jan-2019 (PDF)

- **[Detection of unanticipated faults for autonomous underwater vehicles using online topic models]**

First published: 26 December 2017

https://doi.org/10.1002/rob.21771

https://onlinelibrary.wiley.com/doi/epdf/10.1002/rob.21771

El monitoreo de los parámetros puede servir para detectar con anticipación posibles fallos futuros. Las máquinas de estados finitos pueden realizar un trabajo muy útil en ese ámbito.

- **[A Rule-Based Reasoner for Underwater Robots Using OWL and SWRL]**

  https://www.mdpi.com/1424-8220/18/10/3481/htm

(This article belongs to the Special Issue Underwater Sensor Networks: Applications, Advances and Challenges)

Anexo V: AVISO DE RESPONSABILIDAD

AVISO DE RESPONSABILIDAD:

Este documento es el resultado del Trabajo Fin de Máster de un alumno, siendo su autor responsable de su contenido.

Se trata por tanto de un trabajo académico que puede contener errores detectados por el tribunal y que pueden no haber sido corregidos por el autor en la presente edición.

Debido a dicha orientación académica no debe hacerse un uso profesional de su contenido.

Este tipo de trabajos, junto con su defensa, pueden haber obtenido una nota que oscila entre 5 y 10 puntos, por lo que la calidad y el número de errores que puedan contener difieren en gran medida entre unos trabajos y otros,

La Universidad de Cantabria, la Escuela Técnica Superior de Náutica, los miembros del Tribunal de Trabajos Fin de Máster así como el profesor/a director no son responsables del contenido último de este Trabajo.